

# Trust Region Policy Optimization for POMDPs

Kamyar Azizzadenesheli<sup>1</sup> Manish Kumar Bera<sup>2</sup> Animashree Anandkumar<sup>3</sup>

## Abstract

We propose Generalized Trust Region Policy Optimization (GTRPO), a policy gradient Reinforcement Learning (RL) algorithm for both Markov decision processes (MDP) and Partially Observable Markov Decision Processes (POMDP). Policy gradient is a class of model-free RL methods. Previous policy gradient methods are guaranteed to converge only when the underlying model is an MDP and the policy is run for an infinite horizon. We relax these assumptions to episodic settings and to partially observable models with memory-less policies. For the latter class, GTRPO uses a variant of the Q-function with only three consecutive observations for each policy updates, and hence, is computationally efficient. We theoretically show that the policy updates in GTRPO monotonically improve the expected cumulative return and hence, GTRPO has convergence guarantees.

## 1. Introduction

One of the central challenges in reinforcement learning is the design of efficient algorithms for high-dimensional environments. Recently, Deep-Q networks (Mnih et al., 2015) and its variants, as value-based model-free methods, have shown promise in scaling to large observational spaces. However, these methods are limited to MDPs and mainly dedicated to finite action spaces. Policy gradient methods (Aleksandrov et al., 1968) are another class of model-free methods with no model assumption, therefore conventional approach for continuous high-dimensional action spaces and more importantly for partially observable environments.

Policy gradient approaches mainly deploy Monte Carlo sampling for the gradient update but suffer from high variance gradient estimation (Rubinstein, 1969). To mitigate the high variance shortcoming, recent works deploy value-based methods to the gradient estimation and provide low variance

Table 1: Category of most RL problems

Observability	Policy Class	Horizon	Discounting
MDP	Memory-less	Infinite	Discounted
POMDP	Memory dependent	Episodic	Undiscounted

policy gradient methods (Schulman et al., 2015; Lillicrap et al., 2015). However, they mainly assume the underlying environment is a Markov decision process (MDP), the policy is run to the infinite horizon, and the rewards are undiscounted (Schulman et al., 2015). In practice, many of these assumptions do not hold. The real-world problems are mainly partially observable, episodic and sometimes, the rewards are discounted. It is worth noting that even the empirical study provided in these previous works are episodic, while the theory assumes infinite horizon. If the underlying model of the environment is POMDP, applying MDP based method might result in policies with arbitrarily bad expected returns (Azizzadenesheli et al., 2017).

Table 1 categorizes the majority of RL problems concerning their observability level, policy class, horizon length, and discount factor. Prior methods mainly focus on the memory-less, infinite horizon, undiscounted MDPs. In this work, we focus on episodic MDPs and POMDP with memoryless policies. We investigate both discounted and undiscounted reward settings.

Generally, on-policy policy gradient methods collect data under the policies at hand (current policy) and exploit the acquired data to search for a new and potentially a better policy to deploy. The hope is that this procedure iteratively reinforces the algorithm’s behavior and improves its expected return. Therefore, one of the central goals in policy gradient methods is to develop low variance policy updates which result in the monotonic improvements of the expected returns: the so-called Monotonic Improvement guarantee. Under the infinite horizon undiscounted setting with MDP modeling assumption, Kakade & Langford (2002); Schulman et al. (2015) study the trust-region methods, e.g., TRPO, a class of policy gradients methods which perform the policy search in the trust region around the current policy. They construct a surrogate objective using advantage functions and propose a low variance policy gradient updates. They prove that their low variance policy updates monotonically improves the expected return.

<sup>1</sup>University of California, Irvine, [kazizzard@uci.edu](mailto:kazizzard@uci.edu)

<sup>2</sup>IIT Kanpur, [bera.manish.kumar@gmail.com](mailto:bera.manish.kumar@gmail.com) <sup>3</sup>Caltech, [anima@caltech.edu](mailto:anima@caltech.edu)

In the low sample setting, the accurate estimation of the trust regions is not tractable. TRPO requires to explicitly estimate the trust region to constrain the parameter space which may be hard to maintain in high dimensional and low samples settings. To mitigate this limitation, (Schulman et al., 2017) offer Proximal Policy Optimization (PPO), a simple extension to TRPO, which approximately retains the trust-region constraints directly on the policy space than the parameter space. It also significantly reduces the computation cost of TRPO, therefore it is a reasonable choice for empirical study.

**Contributions:** In this work, we extend the trust region methods, e.g., TRPO, PPO, to episodic MDPs. We show that deploying infinite horizon methods for episodic problem results in a biased estimation of the trust region. We show that it is necessary to incorporate the length of each episode to construct the trust region and extend TRPO and PPO to episodic MDPs.

In presence of discount factor, it is intuitive that the later parts of an episode would have less contribution towards constructing the trust region than the earlier parts. This is also not captured in previous trust region works, e.g. TRPO, PPO. We further extend our analysis and introduce a new notion of distribution divergence, as a discounted sum of Kullback–Leibler (KL) divergences, and show how to construct trust regions in discounted reward settings.

Mainly, the optimal policies for MDPs are deterministic and memoryless. In contrast, we might consider a class of history-dependent policies when we deal with POMDPs. However, tackling history dependent policies can be computationally undecidable (Madani et al., 1999) or PSPACE-Complete (Papadimitriou & Tsitsiklis, 1987), and hence, many works consider the class of stochastic memoryless policies (Azizzadenesheli et al., 2016). In this work, to avoid the computation intractability of history dependent policies, we focus on the class of memory-less policies. Generally, the optimal policies of POMDPs in the class of memoryless policies are indeed stochastic. It is also worth noting that extending the value-based methods through Bellman equations to stochastic memoryless or limited memory policies is not possible if optimality is concerned.

Despite the MDP assumption in the mainstream policy gradient works, empirical studies have demonstrated superior performance when the classes of stochastic policies are considered, e.g., TRPO. The stochastic policies are also known to contribute to the exploration. Many policy gradient algorithms mainly do not converge to deterministic policies in the evaluation period, which is another piece of evidence on partial observability of the environments. Moreover, Sutton et al. (1998) argues that when function approximation methods are deployed to represent the states, due to loss of information in the representation function, the problem

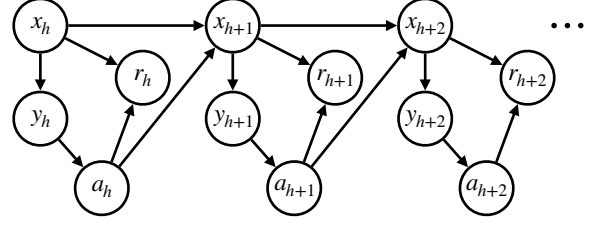


Figure 1: POMDP under a memory-less policy

inherently is a POMDP.

We propose GTRPO, a policy gradient algorithm for episodic POMDPs. GTRPO deploys three consecutive observations in order to approximate an advantage function and computes a low variance estimation of the policy gradient. Surprisingly, deploying three consecutive observations matches the statement in (Azizzadenesheli et al., 2016) which shows statistics of three consecutive observations are necessary to learn the POMDP dynamics and guarantee a regret upper bound in the model-based RL. We construct a trust region for the policy search in GTRPO and show that the policy updates are guaranteed to monotonically improve the expected return. To the best of our knowledge, GTRPO is the first algorithm with monotonic improvement guarantee for the class of POMDP problems.

For the experimental study of GTRPO, we deploy the same methodology used in PPO to reduce the computation cost in TRPO. We apply GTRPO on a variety of RoboSchool (Schulman et al., 2017) environments, which are the extension to the MuJoCo environments (Todorov et al., 2012). We empirically show that despite the computational complexity introduced by most of POMDP based methods, computation complexity introduced by GTRPO is in the same order as its MDP based predecessors. We study GTRPO performance on these simulated environments (RoboSchool environments are almost MDPs, and we do not aim to outperform MDP based methods in these experiments) and report its behavior under different simulation design choices. Throughout the experiments, we observe a similar behavior of the MDP based approach PPO and POMDP based approach GTRPO.

## 2. Preliminaries

An episodic POMDP  $M$  is a tuple  $\langle \mathcal{X}, \mathcal{A}, \mathcal{Y}, P_0, T, R, O, \gamma, x_T \rangle$  with latent state space  $\mathcal{X}$ , observation space  $\mathcal{Y}$ , action space  $\mathcal{A}$ , discount factor of  $0 \leq \gamma \leq 1$  and stochastic reward distribution of  $R(x, a)$  with mean  $\bar{R}(x, a) = \mathbb{E}[R(x, a)]$ ,  $\forall x \in \mathcal{X}, a \in \mathcal{A}$ . Let  $x_T$  denote the terminal state which is *accessible* from any other state, i.e. starting from any other state there is a nonzero probability of reaching the  $x_T$  in finite time steps. The episode terminates when the process reaches  $x_T$ . The initial latent states are drawn from

distribution  $P_1$ . The state dynamics follows transition density  $T(x'|x, a)$ ,  $\forall x, x' \in \mathcal{X}, a \in \mathcal{A}$  and the observation process is generated using density  $O(y|x)$ ,  $\forall x \in \mathcal{X}, y \in \mathcal{Y}$  where a memory-less policy is deployed Fig. 1.

We consider a set of continuously differentiable parameterized memory-less policies  $\pi_\theta$  with  $\theta \in \Theta$ . For each  $y, a$  pair,  $\pi_\theta(a|y)$  denotes the conditional probability distribution of choosing action  $a$  under the policy  $\pi_\theta$  when an observation  $y$  is made. Furthermore, we define a random trajectory  $\tau$  as a finite length  $|\tau|$  sequence of events  $\{(x_1, y_1, a_1, r_1), (x_2, y_2, a_2, r_2), \dots, (x_{|\tau|}, y_{|\tau|}, a_{|\tau|}, r_{|\tau|})\}$  where the termination happens at the step after  $x_{|\tau|}$ , i.e.  $x_{|\tau|+1} = x_T$ . Let  $f(\tau; \theta)$ ,  $\forall \tau \in \Upsilon$  denote the probability distribution of trajectories under policy  $\pi_\theta$  and  $\Upsilon$  is the set of all possible trajectories. Furthermore,  $R(\tau)$  denotes the cumulative  $\gamma$ -discounted rewards of the trajectory  $\tau \in \Upsilon$ . The agent goal is to maximize the unnormalized expected cumulative return  $\eta(\theta) = \mathbb{E}_{\tau|\theta}[R(\tau)]$ ;

$$\theta^* = \arg \min_{\theta \in \Theta} \eta(\theta) := \int_{\tau \in \Upsilon} f(\tau; \theta) R(\tau) d\tau \quad (1)$$

with  $\pi^* = \pi(\theta^*)$  the optimal policy.

### 3. Policy Gradient

In this section, we study the policy gradients methods for POMDPs. Generally, the optimization problem in Eq. 1 is a non-convex problem. Therefore, hill climbing methods such as gradient ascent based approaches might converge to the first order stationary points. Gradient ascent for Eq. 1 results in the policy gradient method. The policy gradient lemma states that the gradient of the expected cumulative return does not require the explicit knowledge of the dynamics but just the cumulative reward distribution (Rubinstein, 1969; Williams, 1992; Baxter & Bartlett, 2001). This lemma has mainly been proven through the construction of score function (see section A.1). In this section, we re-derive the same Lemma but through importance sampling since it is more related to the latter parts of this paper.

Importance sampling is a general technique for estimating the properties of a particular distribution, while only having samples generated from another distribution. One can estimate  $\eta(\theta')$ ,  $\theta' \in \Theta$ , while the expectation is over the distribution induced by  $\pi_\theta$ ;

$$\begin{aligned} \eta(\theta') &= \mathbb{E}_{\tau|\theta'} [R(\tau)] = \int_{\tau \in \Upsilon} f(\tau; \theta) \left( \frac{f(\tau; \theta')}{f(\tau; \theta)} R(\tau) \right) d\tau \\ &= \mathbb{E}_{\tau|\theta} \left[ \frac{f(\tau; \theta')}{f(\tau; \theta)} R(\tau) \right] \end{aligned} \quad (2)$$

as long as for each  $\tau$  that  $f(\tau; \theta') > 0$  also  $f(\tau; \theta) > 0$ .

The gradient of  $\eta(\theta')$  with respect to  $\theta'$  is

$$\begin{aligned} \nabla_{\theta'} \eta(\theta') &= \mathbb{E}_{\tau|\theta} \left[ \frac{\nabla_{\theta'} f(\tau; \theta')}{f(\tau; \theta)} R(\tau) \right] \\ &= \mathbb{E}_{\tau|\theta} \left[ \frac{f(\tau; \theta')}{f(\tau; \theta)} \nabla_{\theta'} \log(f(\tau; \theta')) R(\tau) \right] \end{aligned}$$

The gradient at  $\theta' = \theta$  is;

$$\nabla_{\theta'} \eta(\theta')|_{\theta'=\theta} = \mathbb{E}_{\tau|\theta} [\nabla_{\theta} \log(f(\tau; \theta)) R(\tau)] \quad (3)$$

Since for each trajectory  $\tau$ , the  $\log(f(\tau; \theta))$ ;

$$\begin{aligned} \log \left( P_1(x_1) O(y_1|x_1) R(r_1|x_1, a_1) \prod_{h=2}^{|\tau|} T(x_h|x_{h-1}, a_{h-1}) \right. \\ \left. O(y_h|x_h) R(r_h|x_h, a_h) \right) + \log \left( \prod_{h=1}^{|\tau|} \pi_\theta(a_h|y_h) \right) \end{aligned}$$

and the first part is independent  $\theta$  we have;

$$\nabla_{\theta} \log(f(\tau; \theta)) = \nabla_{\theta} \log \left( \prod_{h=1}^{|\tau|} \pi_\theta(a_h|y_h) \right)$$

This derivation suggest that given trajectories under a policy  $\pi_\theta$  we can compute the gradient of the expected return with respect to the parameters of  $\pi_\theta$  without the knowledge of the dynamics. In practice, however we are not able to compute the exact expectation. Instead we can deploy Monte Carlo sampling technique to estimate the gradient. Given  $m$  trajectories  $\{\tau^1, \dots, \tau^m\}$  with elements  $(x_h^t, y_h^t, a_h^t, r_h^t), \forall h \in \{1, \dots, |\tau^t|\}$  and  $\forall t \in \{1, \dots, m\}$  generated under a policy  $\pi_\theta$ , we can estimate the gradient in Eq. 3 at point  $\theta$ ;

$$\nabla_{\theta} \hat{\eta}(\theta) = \frac{1}{m} \sum_{t=1}^m \nabla_{\theta} \log \left( \prod_{h=1}^{|\tau^t|} \pi_\theta(a_h^t|y_h^t) \right) R(\tau^t) \quad (4)$$

which returns a high variance estimation of the gradient.

#### 3.1. Natural Policy Gradient

Generally, the notion of gradient depends on the parameter metric space. Given a pre-specified Riemannian metric, a gradient direction is defined. When the metric is Euclidean, the notion of gradient reduces to the standard gradient (Lee, 2006). This general notion of gradient adjusts the standard gradient direction based on the local curvature induced by the Riemannian manifold of interest. Valuable knowledge of the curvature assists to find an ascent direction which might conclude to big ascend in the objective function. This approach is also interpreted as a trust region method where we are interested in assuring that the ascent steps do not change the objective beyond a safe region where the local curvature might not stay valid. In general, a valuable manifold might not be given, and we need to adopt one. Fortunately, when the objective function is an expectation over a parameterized distribution, Amari (2016) recommends employing a

Riemannian metric, induced by the Fisher information. This choice of metric results in a well known notion of gradient, so-called *natural gradient*. For the objective function in 1, the Fisher information matrix is defined as follows;

$$F(\theta) := \int_{\tau \in \Upsilon} f(\tau; \theta) \left[ \nabla_{\theta} \log(f(\tau; \theta)) \nabla_{\theta} \log(f(\tau; \theta))^{\top} \right] d\tau \quad (5)$$

Natural gradients are firstly deployed by Kakade (2002) for RL in MDPs. Consequently, the direction of the gradient with respect to  $F$  is defined as  $F(\theta)^{-1} \nabla_{\theta}(\eta(\theta))$ . One can compute the inverse of this matrix to come up with the direction of the natural gradient. Since neither storing the Fisher matrix is always possible nor computing the inverse is practical, direct utilization of  $F(\theta)^{-1} \nabla_{\theta}(\eta(\theta))$  is not feasible. Similar to the approach in TRPO, we suggest to first deploy  $\mathcal{D}_{KL}$  divergence substitution technique and then conjugate gradient method to tackle the computation and storage bottlenecks.

**Lemma 1.** *Under some regularity conditions;*

$$\nabla_{\theta'}^2 \mathcal{D}_{KL}(\theta, \theta')|_{\theta'=\theta} = F(\theta) \quad (6)$$

with  $\mathcal{D}_{KL}(\theta, \theta') := - \int_{\tau \in \Upsilon} f(\tau; \theta) \log(f(\tau; \theta')/f(\tau; \theta)) d\tau$

The Lemma 1 is a known lemma in the literature and we provide its proof in the Subsection A.2. In practice, it is not feasible to compute the expectation in neither the Fisher information matrix nor in the  $\mathcal{D}_{KL}$  divergence, but rather their empirical estimates. Given  $m$  trajectories

$$\begin{aligned} \nabla_{\theta'}^2 \hat{\mathcal{D}}_{KL}(\theta, \theta')|_{\theta'=\theta} &= -\frac{1}{m} \nabla_{\theta'}^2 \sum_{t=1}^m \left[ \log \left( \Pi_{h=1}^{|\tau^t|} \pi_{\theta'}(a_h^t | y_h^t) \right) \right. \\ &\quad \left. \log \left( \Pi_{h=1}^{|\tau^t|} \pi_{\theta}(a_h^t | y_h^t) \right) \right] |_{\theta'=\theta} \\ &= -\frac{1}{m} \nabla_{\theta'}^2 \sum_{t=1}^m \sum_{h=1}^{|\tau^t|} \log \left( \frac{\pi_{\theta'}(a_h^t | y_h^t)}{\pi_{\theta}(a_h^t | y_h^t)} \right) \end{aligned}$$

This derivation of  $\mathcal{D}_{KL}$  is common between MDPs and POMDPs. The analysis in most of the state-of-the-art policy gradient methods, e.g. TRPO, PPO, are dedicated to infinite horizon MDPs, while almost all the experimental studies are in the episodic settings. Therefore the estimator used in these methods;

$$\begin{aligned} \nabla_{\theta'}^2 \hat{\mathcal{D}}_{KL}^{TRPO}(\theta, \theta')|_{\theta'=\theta} &= -\frac{1}{\sum_t |\tau^t|} \nabla_{\theta'}^2 \sum_{t=1}^m \sum_{h=1}^{|\tau^t|} \log \left( \frac{\pi_{\theta'}(a_h^t | y_h^t)}{\pi_{\theta}(a_h^t | y_h^t)} \right) \end{aligned}$$

is a bias estimation of the  $\mathcal{D}_{KL}$  in episodic settings.

### 3.2. $\mathcal{D}_{KL}$ vs $\mathcal{D}_{KL}^{TRPO}$

The use of  $\mathcal{D}_{KL}$  instead of  $\mathcal{D}_{KL}^{TRPO}$  is motivated by theory and also intuitively recommended. A small change in the policy at the beginning of a short episodes does not make a drastic shift in the distribution of the trajectory but might cause radical shifts when the trajectory length is long. Therefore, for longer horizons, the trust region needs to shrink. Consider two trajectories, one long and one short. The  $\mathcal{D}_{KL} \leq \delta$  induces a region which allows higher changes in the policy for short trajectory while limiting changes in long trajectory. While  $\mathcal{D}_{KL}^{TRPO} \leq \delta$  induces the region which does not convey the length of trajectories and looks at each sample as it experienced in a stationary distribution of an infinite horizon MDP.

Consider a toy RL problem where at the beginning of the learning, when the policy is not good, the agent dies at early stages of the episodes (termination). In this case, the trust region under  $\mathcal{D}_{KL}$  is vast and allows for substantial change in the policy space, while again  $\mathcal{D}_{KL}^{TRPO}$  does not consider the length of the episode. On the other hand, toward the end of the learning, when the agent has learnt a good policy, the length of the horizon grows, and small changes in the policy might cause drastic changes in the trajectory distribution. Therefore the trust region shrinks again, and just a small changes in the policy space are allowed, which is again captured by  $\mathcal{D}_{KL}$  but not by  $\mathcal{D}_{KL}^{TRPO}$ .

**Compatible Function Approximation** As it is mentioned before, one way of computing the direction of the natural gradient is to estimate the  $\hat{\mathcal{D}}_{KL}$  and use conjugate gradient methods to find  $F^{-1} \nabla_{\theta}(\eta)$ . There is also an alternative way to estimate  $F^{-1} \nabla_{\theta}(\eta)$ , which is based on compatible function approximation methods. Kakade (2002) study this approach in the context of MDPs. In the following, we develop this approach for POMDPs. Consider a feature map  $\phi(\tau)$  in a desired ambient space defined on  $\Gamma$ . We approximate the return  $R(\tau)$  via a linear function  $\omega$  on the feature representation  $\phi(\tau)$ , i.e.,

$$\min_{\omega} \epsilon(\omega) \text{ s.t.}; \epsilon(\omega) := \int_{\tau \in \Upsilon} f(\tau, \theta) [\phi(\tau)^{\top} \omega - R(\tau)]^2 d\tau$$

To find the optimal  $\omega$  we take the gradient of  $\epsilon(\omega)$  and set it to zero;

$$0 = \nabla_{\omega} \epsilon(\omega)|_{\omega=\omega^*} = \int_{\tau \in \Upsilon} 2f(\tau, \theta) \phi(\tau) [\phi(\tau)^{\top} \omega^* - R(\tau)] d\tau$$

For the optimality,

$$\int_{\tau \in \Upsilon} f(\tau, \theta) \phi(\tau) \phi(\tau)^{\top} \omega^* d\tau = \int_{\tau \in \Upsilon} f(\tau, \theta) \phi(\tau; \theta) R(\tau) d\tau$$

If we consider the  $\phi(\tau) = \nabla_\theta \log \left( \prod_{h=1}^{|\tau|} \pi_\theta(a_h|y_h) \right)$ , the LHS of this equation is  $F(\theta)\omega^*$ . Therefore

$$F(\theta)\omega = \nabla_\theta \eta(\theta) \implies \omega^* = F(\theta)^{-1} \nabla \rho$$

In practice, depending on the problem at hand, either of the discussed approaches of computing the natural gradient might be applicable. Due to the close relationship between  $\mathcal{D}_{KL}$  and Fisher information matrix Lemma 1 and also the fact that the Fisher matrix is equal to second order Taylor expansion of  $\mathcal{D}_{KL}$ , instead of considering the area  $\|(\theta - \theta')^\top F(\theta - \theta')\|_2 \leq \delta$ , or  $\|(\theta - \theta')^\top \nabla_{\theta'}^2 \mathcal{D}_{KL}(\theta, \theta')|_{\theta'=\theta} (\theta - \theta')\|_2 \leq \delta$  for construction of the trust region, we can approximately consider  $\mathcal{D}_{KL}(\theta, \theta') \leq \delta/2$ . This relationship between these three approaches in constructing the trust region is used throughout this paper.

#### 4. TRPO for POMDPs

In this section we extend the MDP analysis in Kakade & Langford (2002); Schulman et al. (2015) to POMDPs, propose GTRPO, and derive a guarantee on its monotonic improvement property. We prove the monotonic improvement property using  $\mathcal{D}_{KL}$ . Moreover, we propose a discount factor dependent divergence and provide the monotonic improvement guarantee w.r.t. this new divergence.

The  $\mathcal{D}_{KL}$  divergence and Fisher information matrix in Eq. 6, Eq. 5 do not convey the effect of the discount factor. Consider a setting with a small discount factor  $\gamma$ . In this setting, we do not mind drastic distribution changes in the latter part of episodes. Therefore, we desire to have a even wider trust region and allow bigger changes for later parts of the trajectories. This is a valid intuition and in the following, we re-derive the  $\mathcal{D}_{KL}$  divergence by also incorporating  $\gamma$ . Let  $\tau_{h'}^h$  denote the elements in  $\tau$  from the time step  $h'$  up to the time step  $h$ ; we rewrite  $\eta(\theta)$  as follows;

$$\eta(\theta) = \int_{\tau \in \Upsilon} f(\tau; \theta) R(\tau) d\tau = \int_{\tau \in \Upsilon} \sum_{h=1}^{|\tau|} f(\tau_1^h; \theta) \gamma^h r_h(\tau) d\tau$$

Following the Amari (2016) reasoning for Fisher information of each component of the sum, we derive a  $\gamma$  dependent divergence;

$$\mathcal{D}_\gamma(\pi_\theta, \pi_{\theta'}) = \sum_{h=1}^{\tau_{\max}} \gamma^h \mathcal{D}_{KL}(\tau_1^h \sim f(\cdot; \pi_{\theta'}), \tau_1^h \sim f(\cdot; \pi_\theta)) \quad (7)$$

For some upper bound on the trajectory lengths,  $\tau_{\max}$ . This divergence less penalizes the distribution mismatch in the later part of trajectories. Similarly, taking into account the relationship between KL divergence and Fisher information we have discount factor  $\gamma$  dependent definition of the Fisher

information;

$$F_\gamma(\theta) := \int_{\tau \in \Upsilon} \sum_{h=1}^{\tau_{\max}} \gamma^h f(\tau_1^h; \theta) \left[ \nabla_\theta \log(f(\tau_1^h; \theta)) \nabla_\theta \log(f(\tau_1^h; \theta))^\top \right] d\tau$$

In the following we develop GTRPO monotonic improvement guarantee under both  $\mathcal{D}_\gamma$  and  $\mathcal{D}_{KL}$ .

##### 4.1. Advantage function on the hidden states

Let  $\pi_\theta$ , the *current policy*, denote the policy under which we collect data, and  $\pi_{\theta'}$ , the *new policy*, the policy which we evaluate its performance. Generally, any memory-less policy on the observation space is transferable to a policy on the latent states as follows;  $\pi(a|x) = \int_{y \in \mathcal{Y}} \pi(a|y) O(y|x) dy$  for each pair of  $(x, a)$ . Consider the case where the agent also observes the latent state, i.e. POMDP  $\rightarrow$  MDP. Since the dynamics on the latent states is MDP, we define the advantage function on the latent states. At time step  $h$  of an episode;

$$\begin{aligned} \tilde{A}_\pi(a, x, h) = \\ \mathbb{E}_{x' \sim T(x'|x, a, h)} \left[ r(x, a, h) + \gamma \tilde{V}_\pi(x', h) - \tilde{V}_\pi(x, h) \right] \end{aligned}$$

Where  $\tilde{V}_\pi$  denote the value function of underlying MDP of latent states when a policy  $\pi$  is deployed. For this choice of advantage function we can write;

$$\begin{aligned} \mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})} \left[ \sum_h^{|\tau|} \gamma^h \tilde{A}_{\pi_\theta}(x_h, a_h, h) \right] \\ = \mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})} \left[ \sum_h^{|\tau|} \gamma^h \left[ r(x_h, a_h, h) + \gamma \tilde{V}_{\pi_\theta}(x_{h+1}, h) - \tilde{V}_{\pi_\theta}(x_h, h) \right] \right] \\ = \mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})} \left[ \sum_h^{|\tau|} \gamma^h r_h \right] - \mathbb{E}_{x_0 \sim P_1(x)} \left[ \tilde{V}_{\pi_\theta}(x_0) \right] \\ = \eta(\pi_{\theta'}) - \eta(\pi_\theta) \end{aligned}$$

This equality suggests that if we have the advantage function of the current policy  $\pi_\theta$  and sampled trajectories from  $\pi_{\theta'}$ , we could compute and maximize the improvement in the expected return  $\eta(\pi_{\theta'}) - \eta(\pi_\theta)$  or, potentially, directly just maximize the expected return for  $\pi_{\theta'}$  without incorporating  $\pi_\theta$ . In practice, we do not have sampled trajectories from the new policy  $\pi_{\theta'}$ , rather we have sampled trajectories from the current policy  $\pi_\theta$ . Therefore, we are interested in maximizing the following surrogate objective function since

we can compute it;

$$\begin{aligned} \tilde{L}_{\pi_{\theta}}(\pi_{\theta'}) &:= \eta(\pi_{\theta}) \\ &+ \mathbb{E}_{\tau \sim \pi_{\theta}, a'_h \sim \pi_{\theta'}(a'_h | x_h, h)} \left[ \sum_h^{|T|} \gamma^h \tilde{A}_{\pi_{\theta}}(x_h, a'_h, h) \right] \end{aligned}$$

For infinite horizon MDPs when  $O$  is an identity map, i.e.,  $x_h = y_h$ , Kakade & Langford (2002); Schulman et al. (2015) show that optimizing  $\tilde{L}_{\pi_{\theta}}(\pi_{\theta'})$  over  $\theta'$  can provide an improvement in the expected discounted return. They derive a lower bound on this improvement if the  $\mathcal{D}_{KL}$  between  $\pi_{\theta'}$  and  $\pi_{\theta}$  for all  $x$ 's is bounded. In the following, we extend these analyses to the general class of environments, i.e. POMDPs and show such guarantees are conserved.

Generally, in POMDPs, when classes of memory-less policies are regarded, neither  $Q$  nor  $V$  functions are well-defined as they are for MDP through the Bellman optimality equations. In the following, we define two quantities similar to the  $Q$  and  $V$  in MDPs and for the simplicity we use the same  $Q$  and  $V$  notation for them. The conditional value and  $Q$ -value functions of POMDPs

$$\begin{aligned} V_{\pi}(y_h, h, y_{h-1}, a_{h-1}) &:= \mathbb{E}_{\pi} \left[ \sum_{h'}^H \gamma^{h'} r_{h'} | y_h, y_{h-1}, a_{h-1} \right] \\ Q_{\pi}(y_{h+1}, a_h, y_h, h) &:= \mathbb{E}_{\pi} \left[ \sum_{h'}^H \gamma^{h'} r_{h'} | y_h, y_{h+1}, a_h \right] \quad (8) \end{aligned}$$

For  $h = 0$  we relax the conditioning on  $y_{h-1}$  for  $V_{\pi}$  and simply denote it as  $V_{\pi}(y, 0)$ . Deploying these two quantities, we define the advantage function as follows;

$$\begin{aligned} A_{\pi}(y_{h+1}, a_h, y_h, h, y_{h-1}, a_{h-1}) \\ = Q_{\pi}(y_{h+1}, a_h, y_h, h) - V_{\pi}(y_h, h, y_{h-1}, a_{h-1}) \end{aligned}$$

The relationship between these two quantity is as follows;

$$\begin{aligned} Q_{\pi}(y_{h+1}, a_h, y_h, h) &:= \\ \mathbb{E}_{\pi} [r_h | y_{h+1}, a_h, y_h] &+ \gamma V_{\pi}(y_{h+1}, h+1, a_h, y_h) \end{aligned}$$

Furthermore, we defined the following surrogate objective;

$$\begin{aligned} L_{\pi_{\theta}}(\pi_{\theta'}) &= \eta(\pi_{\theta}) + \mathbb{E}_{\tau \sim \pi_{\theta}, a \sim \pi_{\theta'}(a|y)} \\ &\sum_h^{|T|} \gamma^h A_{\pi_{\theta}}(y_{h+1}, a_h, y_h, h, y_{h-1}, a_{h-1}) \quad (9) \end{aligned}$$

Similar to MDPs, one can compute and maximize this surrogate objective function in Eq. 9 by just having sampled trajectories and advantage function of the current policy  $\pi_{\theta}$ . But the domain of trust region for the policy search stays unknown. In the following section, we present the trust region for POMDPs.

#### noend 1 GTRPO

- 1: Initial  $\pi_{\theta_0}, \epsilon', \delta'$
- 2: Choice of divergence  $\mathcal{D}$ :  $\mathcal{D}_{KL}$  or  $\mathcal{D}_{\gamma}$
- 3: **for** episode = 1 until convergence **do**
- 4:   Estimate the advantage function  $\hat{A}$
- 5:   Construct the surrogate objective  $\hat{L}_{\pi_{\theta_{t-1}}}(\pi_{\theta})$
- 6:   Find the next policy

$$\pi_{\theta_t} = \arg \max_{\theta} L_{\pi_{\theta_{t-1}}}(\pi_{\theta}) \quad , s.t$$

$$\frac{1}{2} \|(\theta - \theta_{t-1})^T \nabla_{\theta'}^2 \mathcal{D}_{\gamma}(\theta_{t-1}, \theta')|_{\theta'=\theta_{t-1}} (\theta - \theta_{t-1})\|_2 \leq \delta'$$

**Reward Structure:** Similar to MDPs where the reward distribution given the current state, current action and the next state is conditionally independent of the rest of the events, we assume that the reward distribution given the current observation, current action and the next observation is conditionally independent of the rest of the events.

Under this structure we have;

**Lemma 2.** *The improvement in expected return,  $\eta(\pi_{\theta'}) - \eta(\pi_{\theta})$  is equal to;*

$$\mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{|T|} \gamma^h A_{\pi_{\theta}}(y_{h+1}, a_h, y_h, h, y_{h-1}, a_{h-1})$$

*Proof of Lemma 2 in Subsection A.3.*

#### 4.2. GTRPO

We propose generalized trust region policy optimization (GTRPO) as a policy gradient algorithm for POMDPs. GTRPO deploys its current policy to compute the advantage function and then maximize the advantage function over its actions in the vicinity of the current policy. This algorithm is almost identical to its predecessor TRPO except instead of maximizing over on observed hidden state dependent advantage function,  $A_{\pi_{\theta}}(a_h, x_h, h)$ , it maximizes over  $A_{\pi_{\theta}}(y_{h+1}, a_h, y_h, h, y_{h-1}, a_{h-1})$  Alg. 1. It is important to note the one can easily turn the current implementations of TRPO to GTRPO by only changing the line of the code corresponding to the advantage function and substitute it with the proposed one. Moreover, if the model is MDP,  $A_{\pi_{\theta}}(y_{h+1}, a_h, y_h, h, y_{h-1}, a_{h-1})$  is equivalent to  $A_{\pi_{\theta}}(x_{h+1}, a_h, x_h)$  where after marginalizing out  $x_{h+1}$  in the expectation we end up with  $A_{\pi_{\theta}}(a_h, x_h)$  and recover TRPO algorithm.

In practice, one can estimate the advantage function  $A_{\pi_{\theta}}(y_{h+1}, y, a, h, y_{h-1}, a_{h-1})$  by approximating  $Q_{\pi_{\theta}}(y_{h+1}, a, y_h, h)$  and  $V_{\pi_{\theta}}(y_h, h, y_{h-1}, a_{h-1})$  using on-policy data of  $\pi_{\theta}$ . Moreover, for  $L_{\pi_{\theta}}(\pi_{\theta'})$  we have;

$$L_{\pi_{\theta}}(\pi_{\theta}) = \eta(\pi_{\theta}), \text{ and } \nabla_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'})|_{\pi_{\theta}=\pi_{\theta}} = \nabla_{\theta} \eta(\pi_{\theta})$$

In the following we show that maximizing  $L_{\pi_{\theta}}(\pi_{\theta'})$  over  $\theta'$  results in a lower bound on the improvement  $\eta(\pi_{\theta'}) - \eta(\pi_{\theta})$  when  $\pi_{\theta}$  and  $\pi_{\theta'}$  are close under  $\mathcal{D}_{KL}$  or  $\mathcal{D}_{\gamma}$  divergence. Lets define the averaged advantage function

$$\bar{A}_{\pi_{\theta}, \pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) = \mathbb{E}_{a \sim \pi_{\theta'}} [A_{\pi_{\theta}}(y_{h+1}, a, y_h, h, y_{h-1}, a_{h-1})]$$

also the maximum span of the averaged advantage function and its discounted sum as follows;

$$\begin{aligned} \epsilon' &= \max_{\tau \in \Upsilon} \bar{A}_{\pi_{\theta}, \pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) \\ \epsilon &= \max_{\tau \in \Upsilon} \sum_h^{\tau} \gamma^h \bar{A}_{\pi_{\theta}, \pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) \end{aligned}$$

**Theorem 1 (Monotonic Improvement Guarantee).** *For two  $\pi_{\theta}$  and  $\pi_{\theta'}$ , construct  $L_{\pi_{\theta}}(\pi_{\theta'})$ , then*

$$\begin{aligned} \eta(\pi_{\theta'}) &\geq L_{\pi_{\theta}}(\pi_{\theta'}) - \epsilon TV(\tau \sim f(\cdot; \pi_{\theta'}), \tau \sim f(\cdot; \pi_{\theta})) \\ &\geq L_{\pi_{\theta}}(\pi_{\theta'}) - \epsilon \sqrt{\frac{1}{2} \mathcal{D}_{KL}(\pi_{\theta'}, \pi_{\theta})}, \\ \eta(\pi_{\theta'}) &\geq L_{\pi_{\theta}}(\pi_{\theta'}) - \epsilon' \sqrt{\mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'})}. \end{aligned}$$

*Proof of Theorem 1 in Subsection A.4.*

The Theorem. 1 recommends optimizing  $L_{\pi_{\theta}}(\pi_{\theta'})$  over  $\pi_{\theta'}$  around the vicinity defined by  $\mathcal{D}_{KL}$  or  $\mathcal{D}_{\gamma}$  divergences. Therefore, given the current policy  $\pi_{\theta}$  we are interested in either of the following optimization:

$$\begin{aligned} \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) - C \sqrt{\mathcal{D}_{KL}(\pi_{\theta}, \pi_{\theta'})} \\ \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) - C' \sqrt{\mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'})} \end{aligned}$$

Where  $C$  and  $C'$  are the problem dependent constants. Similar to TRPO, using  $C$  and  $C'$  as they are might result in tiny changes in the policy. Therefore, for practical purposes, we view them as the knobs to restrict the trust region denoted by  $\delta$ ,  $\delta'$  and turn these optimization problems to constraint optimization problems;

$$\begin{aligned} \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) \quad \text{s.t.} \quad \mathcal{D}_{KL}(\pi_{\theta}, \pi_{\theta'}) \leq \delta \\ \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) \quad \text{s.t.} \quad \mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'}) \leq \delta' \end{aligned}$$

which results in Alg. 1. Taking into account the relationship between the KL divergence and Fisher information, we can also approximate these two optimization up to their second order Taylor expansion of the constraints;

$$\begin{aligned} \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) \quad \text{s.t.} \quad \frac{1}{2} \|(\theta' - \theta)^{\top} F(\theta' - \theta)\|_2 \leq \delta \\ \max_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'}) \quad \text{s.t.} \quad \frac{1}{2} \|(\theta' - \theta)^{\top} F_{\gamma}(\theta' - \theta)\|_2 \leq \delta' \end{aligned}$$

These analyses provide insights into the design similar algorithm as TRPO and PPO for the general class of POMDPs.

## 5. Experiments

**Extension to PPO:** Usually, in high dimensional but low sample setting, constructing the trust region is hard due to high estimation errors. It is even harder especially when the region depends on the inverse of the estimated Fisher matrix or optimizing over the non-convex function of  $\theta'$  with KL divergence constraint. Therefore, trusting the estimated trust region is questionable. While TRPO constructs the trust region in the parameter space, its final goal is to keep the new policy close to the current policy, i.e., small  $\mathcal{D}_{KL}(\pi_{\theta}, \pi_{\theta'})$  or  $\mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'})$ . Proximal Policy Optimization (PPO) is instead proposed to impose the structure of the trust region directly onto the policy space. This method approximately translates the constraints developed in TRPO to the policy space. It penalized the gradients of the objective function when the policy starts to operate beyond the region of trust by setting the gradient to zero.

$$\begin{aligned} \mathbb{E}[\min\{\frac{\pi_{\theta'}(a|x)}{\pi_{\theta}(a|x)} \tilde{A}_{\pi_{\theta}}(a, x), \\ \text{clip}(\frac{\pi_{\theta'}(a|x)}{\pi_{\theta}(a|x)}; 1 - \delta_L, 1 + \delta_U) \tilde{A}_{\pi_{\theta}}(a, x)\}] \end{aligned}$$

We dropped the  $h$  dependency in the advantage function since this approach is for the infinite horizon MDPs. If the advantage function is positive, and the importance weight is above  $1 + \delta_U$  this objective function saturates. When the advantage function is negative, and the importance weight is below  $1 - \delta_L$  this objective function saturates again. In either case, when the objective function saturates, the gradient of this objective function is zero therefore further development in that direction is obstructed. This approach, despite its simplicity, approximates the trust region effectively and substantially reduce the computation cost of TRPO. **Note:** In the original PPO paper  $\delta_U = \delta_L$ .

Following the TRPO, the clipping trick ensures that the importance weight, derived from estimation of  $\mathcal{D}_{KL}$  does not go beyond a certain limit  $|\log \frac{\pi_{\theta'}(a|y)}{\pi_{\theta}(a|y)}| \leq \nu$ , i.e.,

$$1 - \delta_L := \exp(-\nu) \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta}(a|y)} \leq 1 + \delta_U := \exp(\nu) \quad (10)$$

As discussed in the Remark. 3.2 we propose a principled change in the clipping such that it matches Eq. 6 and conveys information about the length of episodes;  $|\log \frac{\pi_{\theta'}(a|y)}{\pi_{\theta}(a|y)}| \leq \frac{\nu}{|\tau|}$ ; therefore for  $\alpha := \exp(\nu)$

$$1 - \delta_L := \alpha^{-1/|\tau|} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta}(a|y)} \leq 1 + \delta_U := \alpha^{1/|\tau|} \quad (11)$$

This change ensures more restricted clipping for longer trajectories, while wider for shorter ones. Moreover, as it is suggested in theorem. 1, and the definition of  $\mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'})$

in Eq. 7, we propose a further extension in the clipping to conduct information about the discount factor. In order to satisfy  $\mathcal{D}_\gamma(\pi_\theta, \pi_{\theta'}) \leq \delta'$ , for a sample at time step  $h$  of an episode we have  $|\log \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)}| \leq \frac{\nu}{|\tau|\gamma^h}$ . Therefore;

$$\begin{aligned} 1 - \delta_L^h &:= \exp\left(-\frac{\nu}{|\tau|\gamma^h}\right) \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \leq 1 + \delta_U^h := \exp\left(\frac{\nu}{|\tau|\gamma^h}\right) \\ &\rightarrow \alpha^{-1/|\tau|} \alpha^{-1/\gamma^h} \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \leq \alpha^{1/|\tau|} \alpha^{1/\gamma^h} \end{aligned} \quad (12)$$

As it is interpreted, for deeper parts in the episode, we make the clipping softer and allow for larger changes in policy space. This means, we are more restricted at the beginning of trajectories compared to the end of trajectories. The choice of  $\gamma$  and  $\alpha$  are critical here. In practical implementation of RL algorithm, as also theoretically suggested by Jiang et al. (2015); Lipton et al. (2016) we usually choose discount factors smaller than the one for depicted in the problem. Therefore, the discount factor we use in practice is much smaller than the true one specially when we deploy function approximation. Therefore, instead of keeping  $\gamma^h$  in Eq. 12, since the true  $\gamma$  in practice is unknown and can be arbitrary close to 1, we substitute it with a maximum value;

$$\begin{aligned} 1 - \delta_L^h &:= \max\{\alpha^{-1/(|\tau|\gamma^h)}, 1 - \beta\} \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \\ &\leq 1 + \delta_U^h := \min\{\alpha^{1/(|\tau|\gamma^h)}, 1 + \beta\} \end{aligned} \quad (13)$$

The modification proposed in series of equations Eq. 10, Eq. 11, Eq. 12, and Eq. 13 provide insight in the use of trust regions in for both MDPs and POMDPs based PPO. The PPO objective for any choice of  $\delta_U^h$  and  $\delta_L^h$  in MDPs is

$$\begin{aligned} \mathbb{E} \left[ \min \left\{ \frac{\pi_{\theta'}(a_h|x_h)}{\pi_\theta(a_h|x_h)} \tilde{A}_{\pi_\theta}(a_h, x_h), \right. \right. \\ \left. \left. \text{clip}\left(\frac{\pi_{\theta'}(a_h|x_h)}{\pi_\theta(a_h|x_h)}; 1 - \delta_L^h, 1 + \delta_U^h\right) \tilde{A}_{\pi_\theta}(a_h, x_h) \right\} \right] \end{aligned} \quad (14)$$

while for POMDPs we have

$$\begin{aligned} \mathbb{E} \left[ \min \left\{ \frac{\pi_{\theta'}(a_h|y_h)}{\pi_\theta(a_h|y_h)} A_{\pi_\theta}(y_{h+1}, a_h, y_h, y_{h-1}, a_{h-1}), \right. \right. \\ \left. \left. \text{clip}\left(\frac{\pi_{\theta'}(a_h|y_h)}{\pi_\theta(a_h|y_h)}; 1 - \delta_L^h, 1 + \delta_U^h\right) \right. \right. \\ \left. \left. A_{\pi_\theta}(y_{h+1}, a_h, y_h, y_{h-1}, a_{h-1}) \right\} \right] \end{aligned} \quad (15)$$

$h$  is encoded in  $x_h$ . In order to make the existing MDP-based PPO suitable for POMDPs we just substitute  $A_{\pi_\theta}(a_h, x_h)$  with  $A_{\pi_\theta}(y_{h+1}, a_h, y_h, y_{h-1}, a_{h-1})$  in the corresponding line. Moreover, as we showed for TRPO, in the case of MDP model, Eq. 15 reduces to Eq. 14.

**RoboSchool, a variant to MuJoCo:** In the experimental study, we first started to analyze the behavior of the plain PPO agent but observe that the environment enforces a short termination which results in significantly short trajectories. We relaxed this hard threshold and analyzed PPO Section C Subsection C.2. We deploy the analysis in Eq. 11 and Eq. 13, apply the suggested changes to the plain PPO and examine its performance in the variety of different parameters and environments Subsection C.3 and Subsection C.4. In Section E, we apply the GTRPO on the variety of different environments and analyze its behavior. As it is provided in the Appendix, along with the mentioned experimental studies, we have done an extensive study on a variety of different settings to present a more detailed understanding of policy gradient methods. Throughout the experiments, we observe a similar behavior of the MDP based approach PPO and POMDP based approach GTRPO. This might be due to the simplicity of the environment as well as the close similarity of current state of environments are close to MDP. Along the course of the experimental study, we realized that the environment set-up and the deployed reward shaping require a critical and detailed modification to make the test-bed suitable for further studies. Section D and Subsection C.3.2.

## 6. Conclusion

In this paper, we propose GTRPO, a trust region policy optimization method for the general class of POMDPs when the reward process given the current observation, current action, and successive observation is conditionally independent of the rest of variables. We develop a new advantage function for POMDPs which depends on three consecutive observation. The dependency on three consecutive observations also matches the claim in Azizzadenesheli et al. (2016) which shows learning the model and minimizing the regret requires modeling three consecutive observations. GTRPO deploys this advantage function to perform the policy updates. We consider memoryless policies and show that each policy update derived by GTRPO is low variance and monotonically improves the expected return. Additionally, we show how to utilize the analyses in this work and extend the infinite horizon MDP based policy gradient methods, TRPO and PPO, to finite horizon MDPs as well as discounted reward setting. Finally, the same way that PPO extends TRPO and make it computationally more efficient, we extend GTRPO analyses and make it computationally more efficient. We implement this extension and empirical study its behavior along with PPO on Roboschool environments.



## Acknowledgments

K. Azizzadenesheli is supported in part by NSF Career Award CCF-1254106 and Air Force FA9550-15-1-0221. A. Anandkumar is supported in part by Microsoft Faculty Fellowship, Google Faculty Research Award, Adobe Grant, NSF Career Award CCF-1254106, and AFOSR YIP FA9550-15-1-0221.

## References

- Aleksandrov, V. M., Sysoyev, V. I., and Shemenewa, V. V. Stochastic optimaization. *Engineering Cybernetics*, 5 (11-16):229–256, 1968.
- Amari, S.-i. *Information geometry and its applications*. Springer, 2016.
- Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1602.07764*, 2016.
- Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Experimental results: Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1705.02553*, 2017.
- Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.
- Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.
- Jiang, N., Kulesza, A., Singh, S., and Lewis, R. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1181–1189, 2015.
- Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.
- Kakade, S. M. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.
- Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr>, 2018.
- Lee, J. M. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., and Deng, L. Combating reinforcement learning’s sisyphian curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.
- Madani, O., Hanks, S., and Condon, A. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pp. 541–548, 1999.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
- Papadimitriou, C. H. and Tsitsiklis, J. N. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.
- Rubinstein, R. Y. Some problems in monte carlo optimization. *Ph.D. thesis*, 1969.
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Sutton, R. S., Barto, A. G., Bach, F., et al. *Reinforcement learning: An introduction*. MIT press, 1998.
- Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.
- Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

## A. Appendix

### A.1. Score function

It is well known that the gradient of the expected cumulative return can be approximated without knowledge of the model dynamics (Williams, 1992; Baxter & Bartlett, 2001). We restate this development of the previous works for POMDPs from the point of view of score function

$$\nabla_{\theta}\eta(\theta) = \nabla_{\theta} \int_{\tau \in \Upsilon} f(\tau; \theta) R(\tau) d\tau = \int_{\tau \in \Upsilon} \nabla_{\theta} f(\tau; \theta) R(\tau) d\tau = \int_{\tau \in \Upsilon} f(\tau; \theta) \nabla_{\theta} \log(f(\tau; \theta)) R(\tau) d\tau$$

for a single trajectory of  $\tau = \{(x_1, y_1, a_1, r_1), (x_2, y_2, a_2, r_2), \dots, (x_{|\tau|}, y_{|\tau|}, a_{|\tau|}, r_{|\tau|})\}$ ,  $R(\tau) = \sum_{h=1}^{|\tau|} r_h$ . while

$$f(\tau; \theta) = P_1(x_1) O(y_1|x_1) \pi_{\theta}(a_1|y_1) R(r_1|x_1, a_1) \prod_{h=2}^{|\tau|} T(x_h|x_{h-1}, a_{h-1}) O(y_h|x_h) \pi_{\theta}(a_h|y_h) R(r_h|x_h, a_h)$$

Therefore, for the gradient of the log we have;

$$\begin{aligned} \nabla_{\theta} \log(f(\tau; \theta)) &= \nabla_{\theta} \log \left( P_1(x_1) O(y_1|x_1) R(r_1|x_1, a_1) \prod_{h=2}^{|\tau|} T(x_h|x_{h-1}, a_{h-1}) O(y_h|x_h) R(r_h|x_h, a_h) \right) \\ &\quad + \nabla_{\theta} \log \left( \prod_{h=1}^{|\tau|} \pi_{\theta}(a_h|y_h) \right) \end{aligned}$$

since the first part is independent of  $\theta$ , its derivative is zero. Therefore we have

$$\nabla_{\theta}\eta(\theta) = \int_{\tau \in \Upsilon} f(\tau; \theta) \nabla_{\theta} \log \left( \prod_{h=1}^{|\tau|} \pi_{\theta}(a_h|y_h) \right) R(\tau) d\tau$$

It is clear through Monte Carlo sampling theorem that given a set of  $m$  trajectories  $\{\tau^1, \dots, \tau^m\}$  with elements  $(x_h^t, y_h^t, a_h^t, r_h^t), \forall h \in \{1, \dots, |\tau^t|\}$  and  $\forall t \in \{1, \dots, m\}$ , the empirical mean of the gradient is

$$\widehat{\nabla}_{\theta}(\eta) = \frac{1}{m} \sum_{t=1}^m \nabla_{\theta} \log \left( \prod_{h=1}^{|\tau^t|} \pi_{\theta}(a_h^t|y_h^t) \right) R(\tau^t) \quad (16)$$

which does not depend on underlying dynamic except through cumulative reward  $R(\tau)$

### A.2. Proof of Lemma 1

*Proof.*

$$\begin{aligned}
 \nabla_{\theta'}^2 KL(\theta, \theta')|_{\theta'=\theta} &:= -\nabla_{\theta'}^2 \int_{\tau \in \Upsilon} f(\tau; \theta) [\log(f(\tau; \theta')) - \log(f(\tau; \theta))] d\tau|_{\theta'=\theta} \\
 &= -\int_{\tau \in \Upsilon} f(\tau; \theta) \nabla_{\theta'}^2 \log(f(\tau; \theta')) d\tau|_{\theta'=\theta} \\
 &= -\int_{\tau \in \Upsilon} f(\tau; \theta) \nabla_{\theta'} \left[ \frac{1}{f(\tau; \theta')} \nabla_{\theta'} f(\tau; \theta') \right] d\tau|_{\theta'=\theta} \\
 &= \int_{\tau \in \Upsilon} f(\tau; \theta) \left[ \frac{1}{f(\tau; \theta')^2} \nabla_{\theta'} f(\tau; \theta') \nabla_{\theta'} f(\tau; \theta')^\top \right] d\tau|_{\theta'=\theta} \\
 &\quad - \int_{\tau \in \Upsilon} f(\tau; \theta) \left[ \frac{1}{f(\tau; \theta')} \nabla_{\theta'}^2 f(\tau; \theta') \right] d\tau|_{\theta'=\theta} \\
 &= \int_{\tau \in \Upsilon} f(\tau; \theta) \left[ \frac{1}{f(\tau; \theta')^2} \nabla_{\theta'} f(\tau; \theta') \nabla_{\theta'} f(\tau; \theta')^\top \right] d\tau|_{\theta'=\theta} \\
 &\quad - \nabla_{\theta'}^2 \int_{\tau \in \Upsilon} f(\tau; \theta') d\tau|_{\theta'=\theta} = F(\theta)
 \end{aligned} \tag{17}$$

□

### A.3. Proof of Lemma 2

*Proof.* With a few substitutions in the first term and the assumption that the reward depends on current action, current observation, and next observation we have;

$$\begin{aligned}
 &\mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h A_{\pi_{\theta}}(y_{h+1}, y_h, a_h, h, y_{h-1}, a_{h-1}) \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h [Q_{\pi_{\theta}}(y_{h+1}, a_h, y_h, h) - V_{\pi_{\theta}}(y_h, h, y_{h-1}, a_{h-1})] \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h [\mathbb{E}_{\pi_{\theta}} [r_h | y_h, y_{h+1}, a_h] + \gamma V_{\pi_{\theta}}(y_{h+1}, h+1, y_h, a_h) - V_{\pi_{\theta}}(y_h, h, y_{h-1}, a_{h-1})] \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h [\mathbb{E} [r_h | y_h, y_{h+1}, a_h] + \gamma V_{\pi_{\theta}}(y_{h+1}, h+1, y_h, a_h) - V_{\pi_{\theta}}(y_h, h, y_{h-1}, a_{h-1})] \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h \mathbb{E} [r_h | y_h, y_{h+1}, a_h] - \mathbb{E} [V_{\pi_{\theta}}(y_0, 0)] \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h \mathbb{E} [r_h | y_h, y_{h+1}, a_h] - \eta(\pi_{\theta}) \\
 &= \eta(\pi_{\theta'}) - \eta(\pi_{\theta})
 \end{aligned}$$

□

### A.4. Proof of Theorem 1

*Proof.* Following the result in the Lemma 2 we have

$$\begin{aligned}
 \eta(\pi_{\theta'}) &= \eta(\pi_{\theta}) \\
 &\quad + \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h A_{\pi_{\theta}}(y_{h+1}, y_h, h, a_h, y_{h-1}, a_{h-1})
 \end{aligned}$$

therefore,

$$\begin{aligned}
 & \eta(\pi_{\theta'}) - L_{\pi}(\pi_{\theta'}) \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \sum_h^{| \tau |} \gamma^h A_{\pi_{\theta}}(y_{h+1}, y_h, h, a_h, y_{h-1}, a_{h-1}) \\
 & - \mathbb{E}_{\tau \sim \pi_{\theta}, a'_h \sim \pi_{\theta'}(a|y)} \sum_h^{| \tau |} \gamma^h A_{\pi_{\theta}}(y_{h+1}, y_h, h, a'_h, y_{h-1}, a_{h-1})
 \end{aligned}$$

following the definition of  $\tilde{A}_{\pi_{\theta}, \pi_{\theta'}}$

$$\begin{aligned}
 \eta(\pi_{\theta'}) - L_{\pi_{\theta}}(\pi_{\theta'}) &= \int_{\tau} (f(\tau; \pi_{\theta'}) - f(\tau; \pi_{\theta})) \\
 & \sum_h^{| \tau |} \gamma^h \bar{A}_{\pi_{\theta}, \pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) d\tau
 \end{aligned}$$

Deploying the maximum span of averaged advantage function  $\epsilon$  and the Pinsker's inequality we have

$$\begin{aligned}
 & |\eta(\pi_{\theta'}) - L_{\pi_{\theta}}(\pi_{\theta'})| \\
 & \leq \epsilon TV(\tau \sim f(\cdot; \pi_{\theta'}), \tau \sim f(\tau; \pi_{\theta})) \\
 & \leq \epsilon \sqrt{\frac{1}{2} \mathcal{D}_{KL}(\tau \sim f(\cdot; \pi_{\theta'}), \tau \sim f(\tau; \pi_{\theta}))}
 \end{aligned}$$

Which results in the first part of the theorem. On the other hand

$$\begin{aligned}
 \eta(\pi_{\theta'}) - L_{\pi_{\theta}}(\pi_{\theta'}) &= \int_{\tau} \sum_{h=1}^{| \tau |} (f(\tau_1^h; \pi_{\theta'}) - f(\tau_1^h; \pi_{\theta})) \\
 & \gamma^h \bar{A}_{\pi_{\theta}, \pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) d\tau
 \end{aligned}$$

Deploying the definition of  $\epsilon'$  and the Pinsker's inequality again we have

$$\begin{aligned}
 & |\eta(\pi_{\theta'}) - L_{\pi_{\theta}}(\pi_{\theta'})| \\
 & \leq \epsilon' \sum_{h=1}^{\tau_{\max}} \gamma^h TV(\tau_1^h \sim f(\cdot; \pi_{\theta'}), \tau_1^h \sim f(\cdot; \pi_{\theta})) \\
 & \leq \epsilon' \sum_{h=1}^{\tau_{\max}} \gamma^h \sqrt{\frac{1}{2} \mathcal{D}_{KL}(\tau_1^h \sim f(\cdot; \pi_{\theta'}), \tau_1^h \sim f(\cdot; \pi_{\theta}))} \\
 & \leq \epsilon' \sqrt{\mathcal{D}_{\gamma}(\pi_{\theta}, \pi_{\theta'})}
 \end{aligned}$$

and the second part of the theorem goes through. □

## B. Experimental Study

In the following sections we empirically study the sequence of changes that the theoretical analyses suggest to make on PPO. The code for PPO that we used can be found <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr> (Kostrikov, 2018). We use <https://blog.openai.com/roboschool/> environments for our experiments.

---

**General and Important Note:** The goal of the experimental study is not to improve the leaderboard of DeepRL scores, and rather a report on the empirical behavior of GTRPO. Since the MuJoCo environments are closely MDP, we also do not expect any improvement over MDP based methods.

We have primarily experimented 4 propositions:

1. **PPO-length- $\gamma$ -dependent:** Study of PPO when we incorporate length dependent and discount factor dependent way of constructing the trust region.
2. **GTRPO:** The study of GTRPO
3. **Environment Choice:** The study of the Roboschool environments
4. **PPO through signSGD:** A further study of the trust region construction through sign gradient methods, e.g., signSGD (Bernstein et al., 2018).

Note: In all of the following plots, unless otherwise mentioned, the x-axis represents the number of steps that the model has seen, and the y-axis represents the reward. The graph has been normalized and made smooth for better visualization. The label of the plots are the name of the corresponding **roboschool** environment.

## C. PPO-length- $\gamma$ -dependent

### C.1. ppo-len-dep

In PPO, in order to provide a better approximation through Monte Carlo sampling, the policy updates take place after one(or more) full episodes of experiences. The motivation of this variation of PPO lies in the intuition that the trust region deployed for policy update should depend on the number of steps in the episode. When the model is in its initial stages of learning, it is most probable that the length of episodes is very low. Therefore more significant changes in the policy space should be allowed. As the RL agent becomes more experienced, it acquires a better policy, deals better with the environment, and thus, the episode lengths increase. At this point, it is essential that the updates stay small since even small changes might result in a drastic shift in the trajectory distributions. Therefore we modify the update as mentioned in Eq. 11

The original implementation of PPO in <https://github.com/ikostrikov> has a fixed number (128 to be exact) for the maximum time-steps per episode. Almost all the models would hit this limit very quickly. Setting a significantly low threshold for episode length reduces the capability to study the behavior of RL algorithms. Therefore we increased the number this maximum threshold to 1000. We denote this PPO on this environment as **ppo-1000steps**. The original variant (vanilla PPO) is referred as **ppo-original**.

### C.2. ppo-original v/s ppo-1000steps

Before proceeding to the experiments with **ppo-1000steps**, we compare the performances of **ppo-1000steps** and **ppo-original**. Fig. 2 provide the mentioned comparison. We observe that changing episode length threshold does not affect the final performance of either of these two by far. We observe that the convergent value is achieved faster in case of **ppo-original**, this might be caused by the fact that **ppo-original** is allocated more number of updates than **ppo-1000steps** for a given number of total time-steps (x-axis).

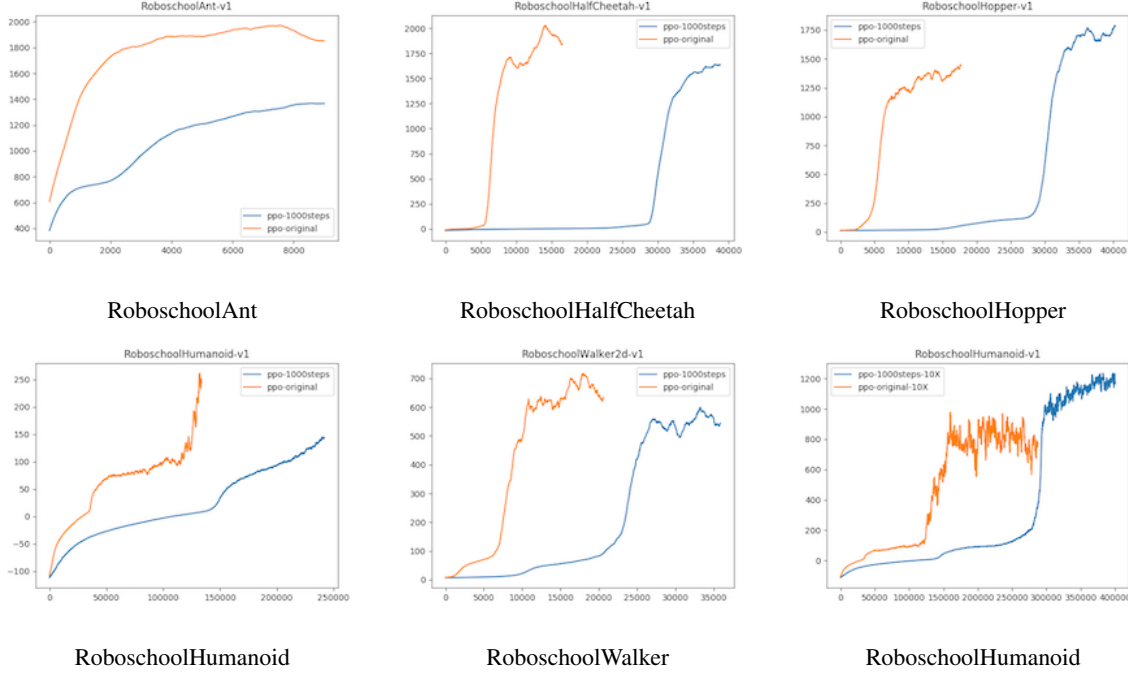


Figure 2: We run PPO on 5 environments with two design choice. **ppo-original** denote the PPO agent with maximum length of each episode is equal to 128. **ppo-1000steps** denotes the PPO agent when the maximum length of each episode is equal to 1000 .

#### C.2.1. PPO-ORIGINAL-10X v/s PPO-1000STEPS-10X

From Fig.2 one can observe that for the environment **humanoid**, **ppo-1000steps** has hard time to converge under the same number of steps. Therefore, we increased the number of episode 10 times and rerun both of them Fig. 3.

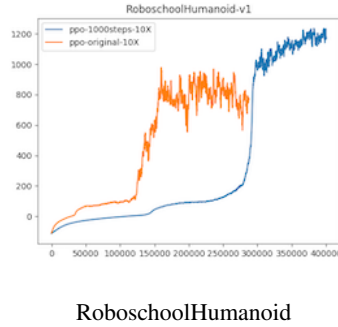


Figure 3: We run PPO on humanoid environment with two design choice but this time for 10 times longer. **ppo-original** denote the PPO agent with maximum length of each episode is equal to 128. **ppo-1000steps** denotes the PPO agent when the maximum length of each episode is equal to 1000 .

#### C.2.2. EPISODE LENGTHS

In this subsection, we study the episode length distribution induced by **ppo-1000steps** on various environments after we set the maximum threshold of per-episode steps to 1000. In Fig. 4 we plot the episode lengths of **ppo-1000steps** as it learns the policy. The x-axis represents the number of episodes seen by the model, and the y-axis represents the episode length..

#### C.2.3. PPO-1000STEPS-10X

Similar as before, we run the humanoid also for 10 times longer time steps. Fig. 5

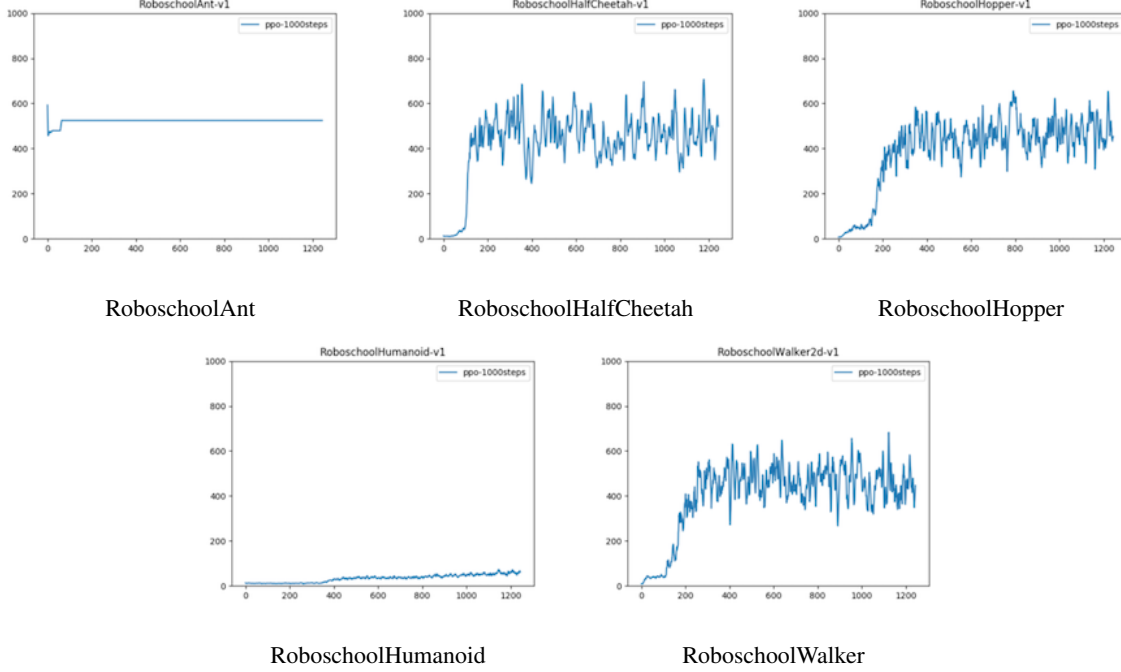


Figure 4: The episode length distribution induced by **ppo-1000steps** on various environments

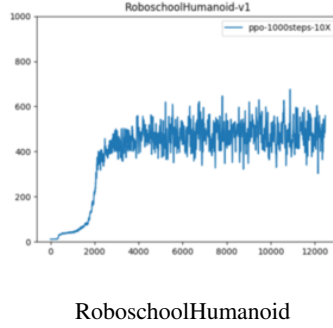


Figure 5: The episode length distribution induced by **ppo-1000steps** on humanoid environment when we run it for 10 times longer

### C.3. ppo-1000steps vs ppo-1000steps-len-dep

As it is discussed in Section 5, the PPO (Schulman et al., 2017) deploys clipping to make the policy updates maximal while conservative, i.e., for some threshold  $\alpha_x$  on states of a MDP

$$\alpha_x \leq \frac{\pi_{\theta'}(a|x)}{\pi_{\theta_\theta}(a|x)} \leq \alpha_x$$

As mentioned in the Eq. 11 for episodic setting, the trust region should depend on the length of trajectories. We re-state the Eq. 11 here

$$1 - \delta_L = \alpha^{-1/|\tau|} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta_\theta}(a|y)} \leq 1 + \delta_U = \alpha^{1/|\tau|}$$

We compare the performances of **ppo-1000steps** and **ppo-1000steps-len-dep** which is same as PPO except the clipping  $\delta_L$  and  $\delta_U$  are defined as in Eq. 11. In Fig. 6, as usual, x-axis represents the number of time steps seen by the model, and y-axis represents rewards at each time step. The legend denotes the values of the  $\alpha$ .

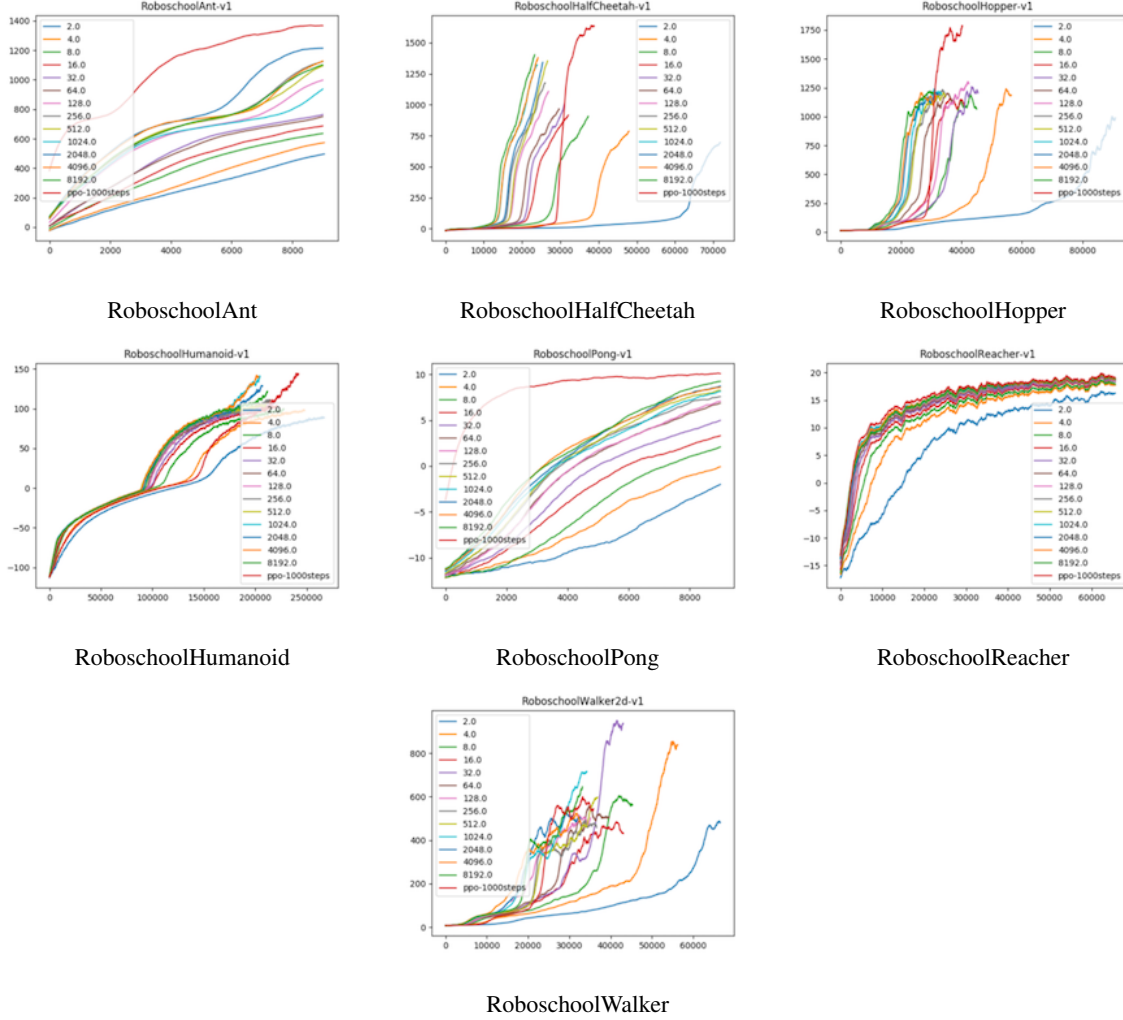


Figure 6: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 11, **ppo-1000steps-len-dep**, and variety of different choices of  $\alpha$

### C.3.1. RUNNING 10X EPISODES FOR HUMANOID

The **humanoid** environment of **roboschool** takes longer to converge than the other environments. Same as before, we run it for 10X more episodes to observe the behaviour and convergent values, Fig. 7



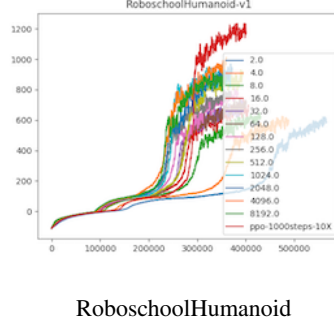


Figure 7: We run PPO on Humanoid environments in Robo-School for 10 times longer than other environments. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 11, **ppo-1000steps-len-dep**, and variety of different choices of  $\alpha$

### C.3.2. EPISODE LENGTH

We further study the episode length to see if there are any changes to how the episode length modulates over the training period when we deploy **ppo-1000steps-len-dep** with a variety of  $\alpha$ 's. The x-axis represents the number of episodes seen by the model, y-axis represents the number of steps in that episode, and the legend represents the value of  $\delta$  (see above). We observe that there is no significant change in the behaviour of episode length.

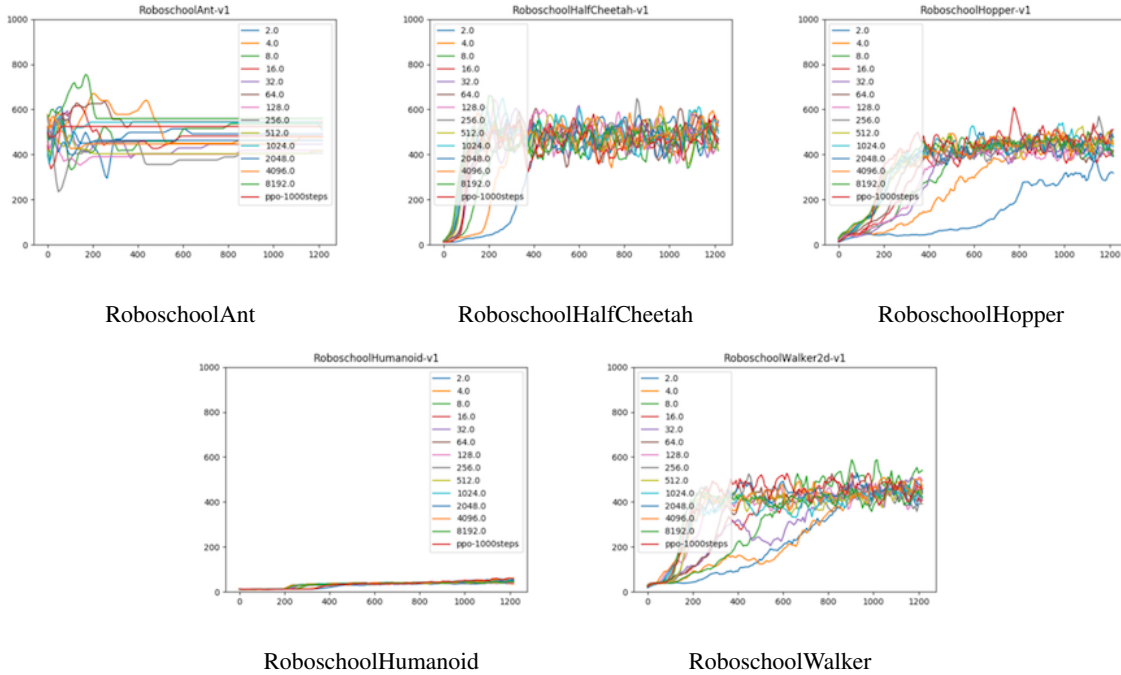


Figure 8: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 11, **ppo-1000steps-len-dep**, and variety of different choices of  $\alpha$ . This figure represents the episode length behaviour over the course of training.

### C.4. ppo-1000steps vs ppo-1000steps-len- $\gamma$ -dep

In the previous section, we study the PPO behaviour when we followed the the  $\mathcal{D}_{KL}$  divergence definition and Eq. 11, i.e., **ppo-1000steps-len-dep**. In this subsection, we study the trust region suggested by  $\mathcal{D}_{\gamma}$  the discount factor dependent trust

region construction, and Eq. 13

$$\max\{\alpha^{-1/(|\tau|\gamma^h)}, 1 - \beta\} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta}(a|y)} \leq 1 + \delta_U^h := \min\{\alpha^{1/(|\tau|\gamma^h)}, 1 + \beta\}$$

We study the behaviour of **ppo-1000steps-len- $\gamma$ -dep**. We set  $1 - \delta_L^h := \max(1 - \beta, (\frac{1}{\alpha})^{\frac{1}{|\tau|\gamma^h}})$  and  $1 + \delta_U^h := \min(1 + \beta, \alpha^{\frac{1}{|\tau|\gamma^h}})$ .

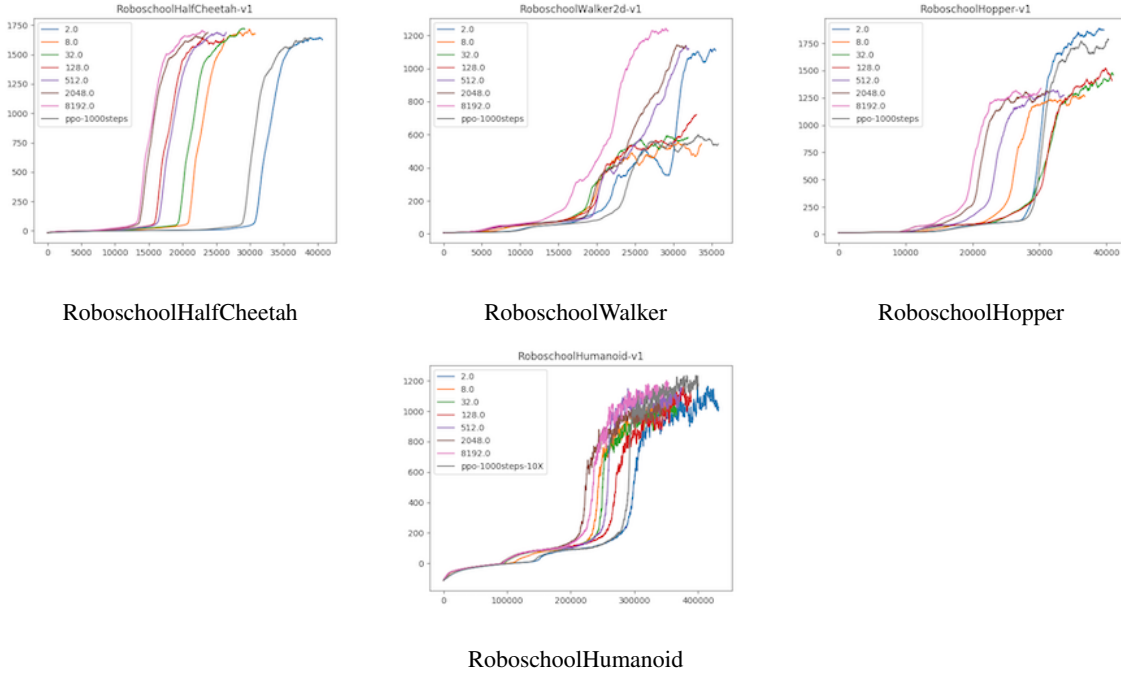


Figure 9: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for **ppo-1000steps-len- $\gamma$ -dep** and variety  $\beta$  choices

### C.5. ppo-1000steps-dynamic-clip

In the original parameters set by Ilya Kostrikov, the total number of frames was set as  $10^6$  while  $\delta_U = \delta_L = 0.1$  and followed by the Eq.10. The clipping parameter of 0.1 is useful for start of training. When a good policy is learnt, making the trust region more conservative and shrinking the clipping parameter to smaller value might be helpful to find better policy. For this study, we first train plain PPO of **ppo-1000steps** with on clip of 0.1 for  $10^6$  frames, and then train it with clipping parameter 0.05 for the next  $10^6$  frames. We express the empirical results for both **ppo-original** with threshold of 128 as the maximum length of episodes, and **ppo-1000steps** with threshold of 1000 as the maximum length of episodes.

#### C.5.1. MAX EPISODE LENGTHS = 1000

The following plots, Fig. 10 are for experiments run with maximum episode length set as 1000 steps. In the legend **ppo-1000steps-2X** denotes running the vanilla **ppo-1000steps** for 2X number of episodes; **ppo-1000steps-dynamic\_clip** denotes model with the above mentioned changes.

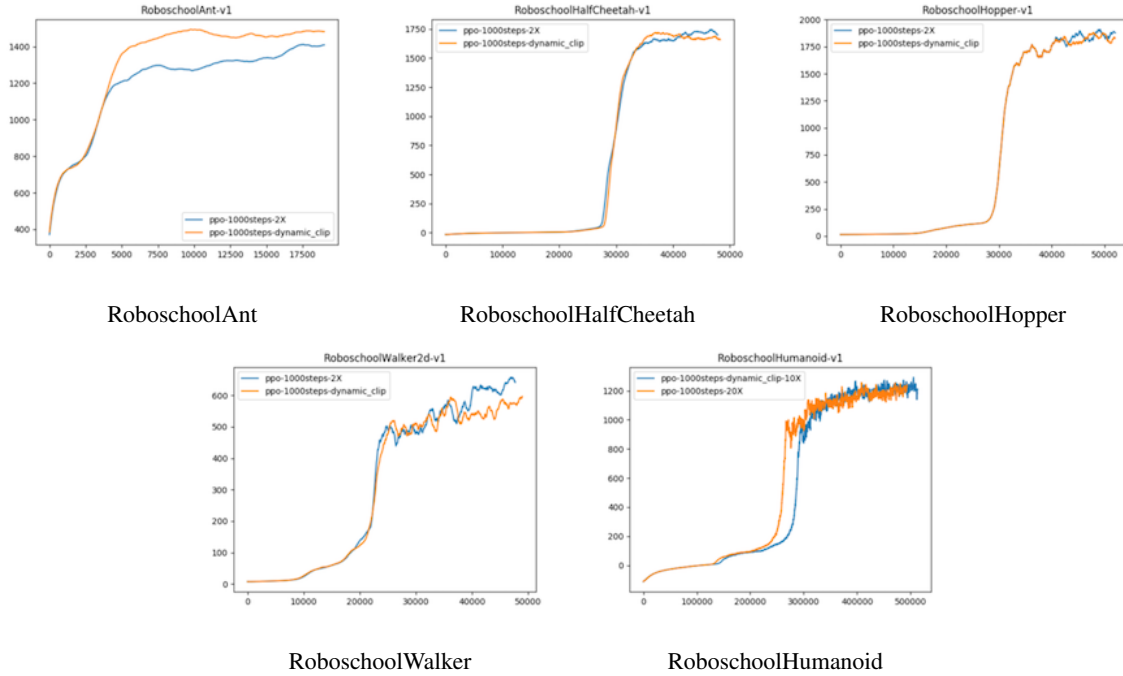


Figure 10: We run **ppo-1000steps** for 2 times longer and denote it as **ppo-1000steps-2X**. We also run **ppo-1000steps-dynamic\_clip** which is **ppo-1000steps-2X** except for the first 10e6 steps the clipping parameter is 0.1 and for the second 10e6 it is set to 0.05

### C.5.2. MAX EPISODE LENGTH = 128

The following plots in Fig. 11 we run the same experiments as Subsection C.5.1 but for threshold on maximum length of episode set to 128 steps.



Figure 11: We run **ppo-original** for 2 times longer and denote it as **ppo-2X**. We also run **ppo-dynamic.clip** which is **ppo-2X** except for the first  $10e6$  steps the clipping parameter is 0.1 and for the second  $10e6$  it is set to 0.05

### C.6. ppo-1000steps-equalizer vs ppo-len- $\gamma$ -dep-equalizer running

In this subsection, we study the same empirical setting as the subsection C.4 but instead of running the algorithms for the same count of episodes we run them for the same number of interactions with the environment. We study the behaviour of **ppo-1000steps-len- $\gamma$ -dep** and **ppo-1000steps-len** when we run both for the same number of time steps and denote them **ppo-1000steps-len- $\gamma$ -dep-equalizer** and **ppo-1000steps-len-equalizer**. We run **ppo-1000steps-len- $\gamma$ -dep-equalizer** for a fixed  $\beta = 0.1$  and also  $\beta = 0.1$  for a variety of  $\alpha$ 's. We experiment these for the cases when the threshold on the maximum length is 1000 as well as 128.

#### C.6.1. $\alpha = 0.1$ AND EPISODE LENGTH = 1000

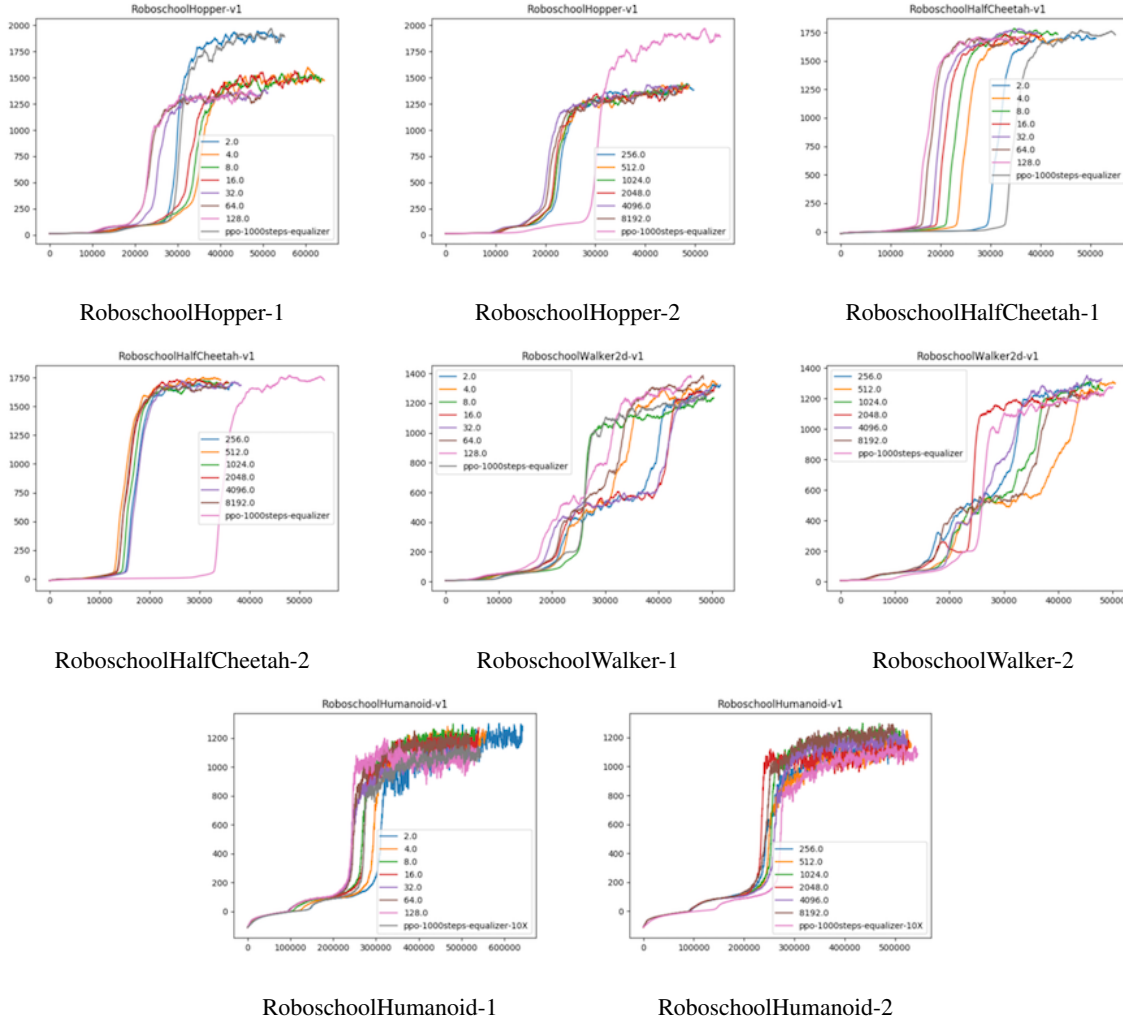


Figure 12: We run **ppo-1000steps** and **ppo-1000steps-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length 1000 and call them **ppo-1000steps-equalizer** and **ppo-1000steps-len- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$

C.6.2. FOR  $\beta = 0.1$  AND EPISODE LENGTH = 128

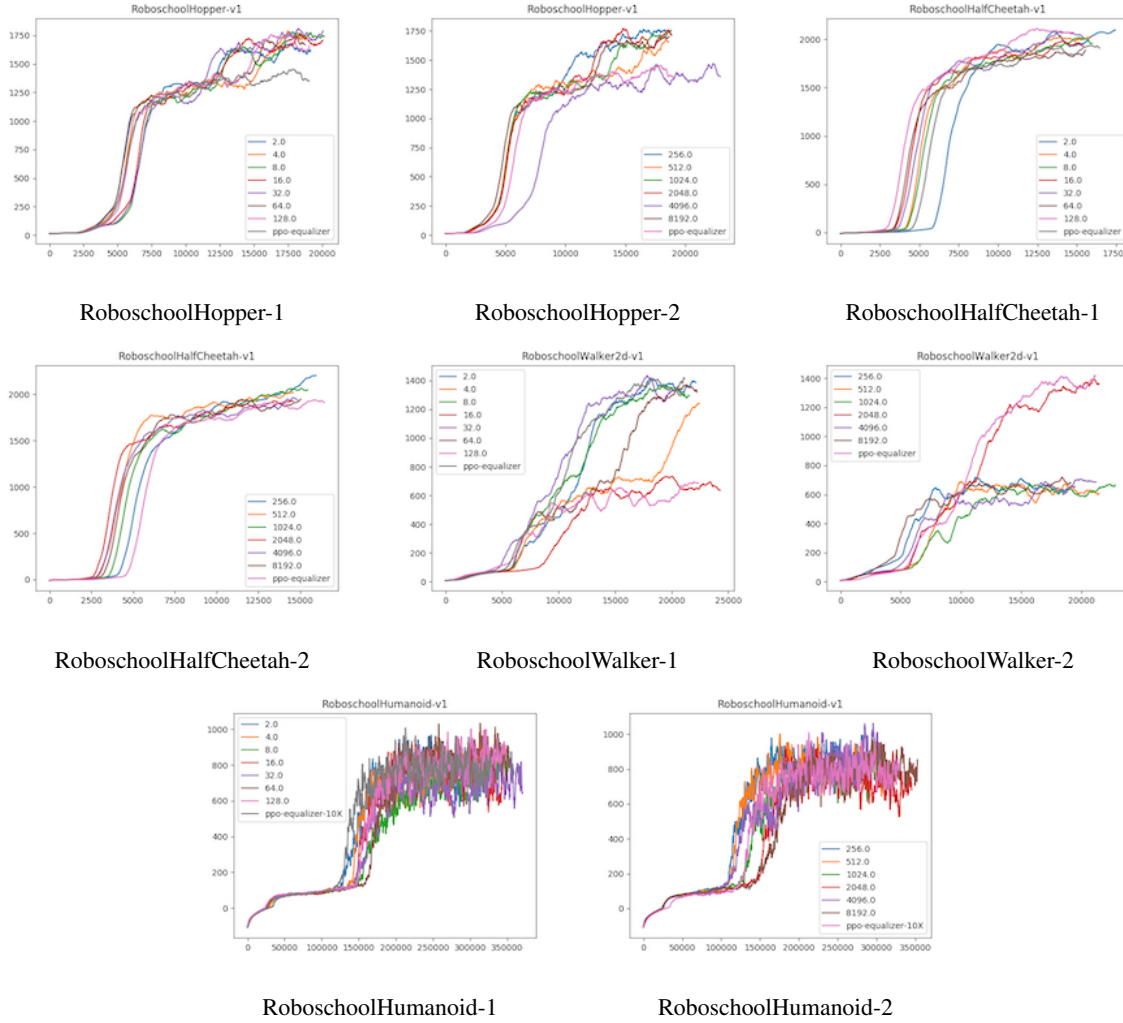


Figure 13: We run **ppo-original** and **ppo-original- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$

C.6.3.  $\beta = 0.05$  AND EPISODE LENGTH = 1000

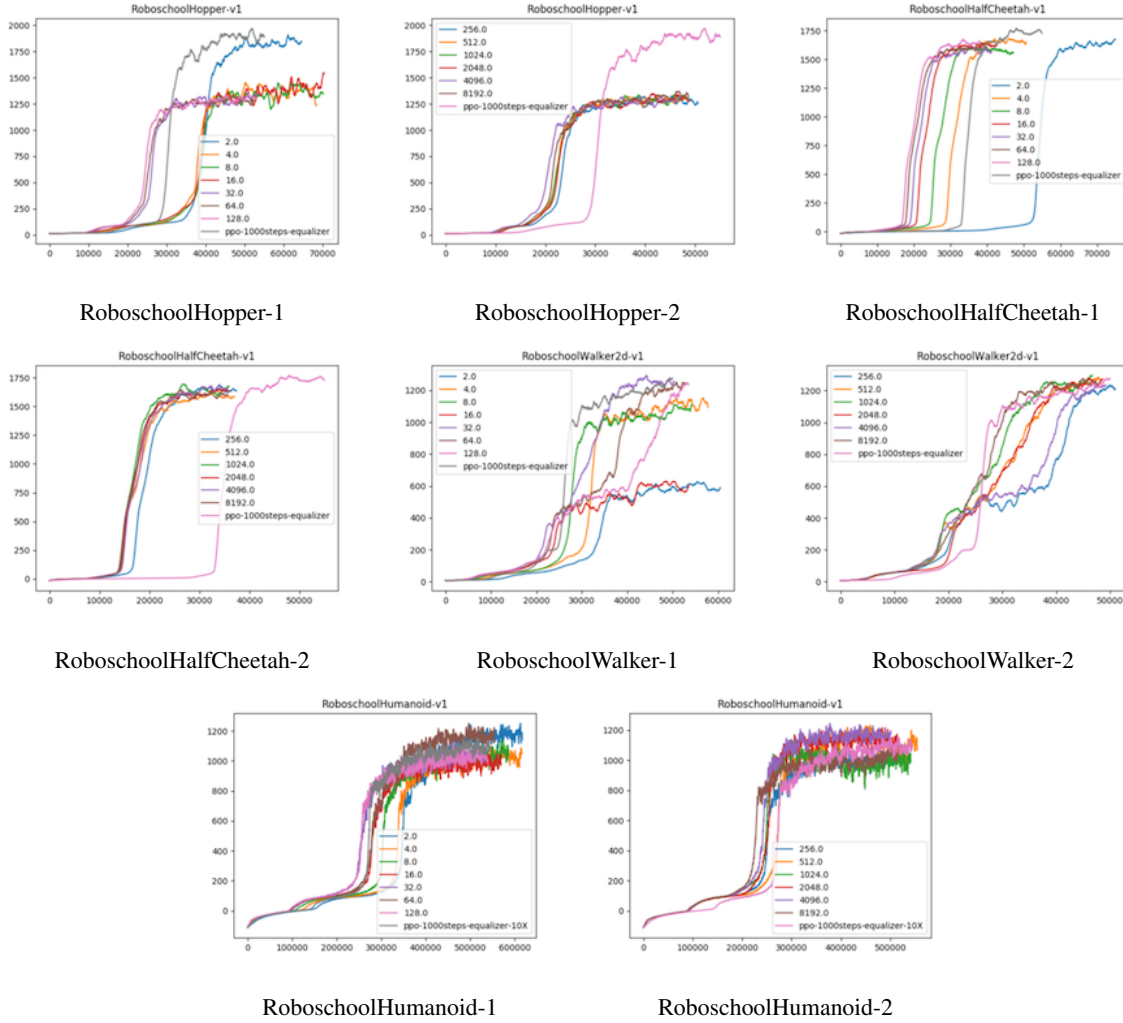


Figure 14: We run **ppo-original** and **ppo-original-gamma-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-gamma-dep-equalizer**. We keep  $\beta = 0.01$  and vary  $\alpha$

C.6.4.  $\beta = 0.05$  AND EPISODE LENGTH = 128

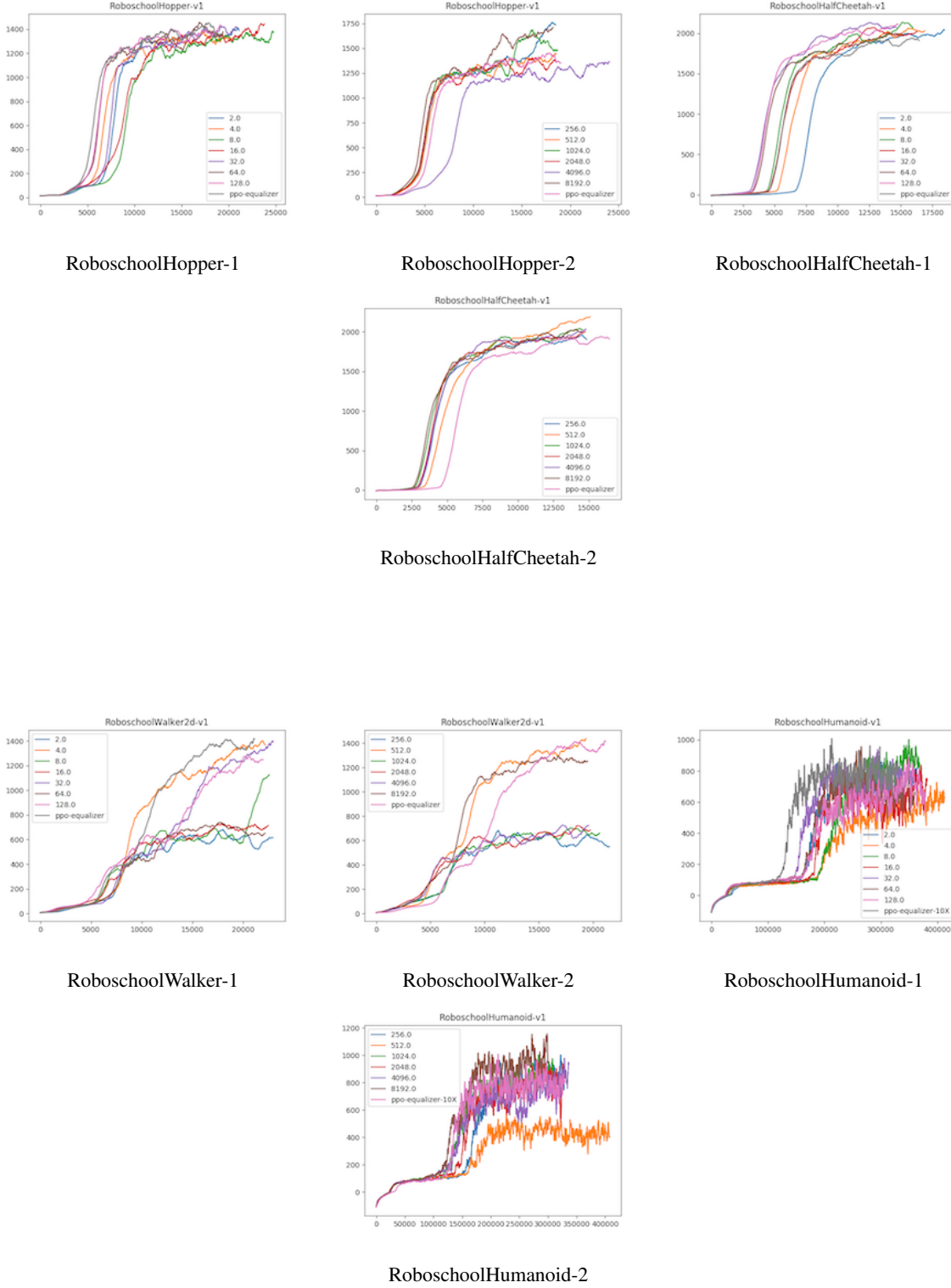


Figure 15: We run **ppo-original** and **ppo-original- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.05$  and vary  $\alpha$



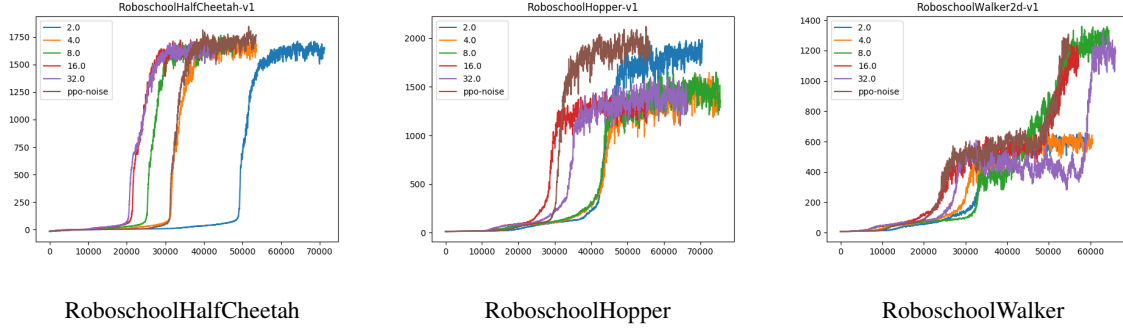
C.7. ppo-1000steps-equalizer vs ppo-len- $\gamma$ -dep-equalizerC.7.1. FOR  $\beta = 0.05$  AND  $stddev = 0.001$ 

Figure 16: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.05$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

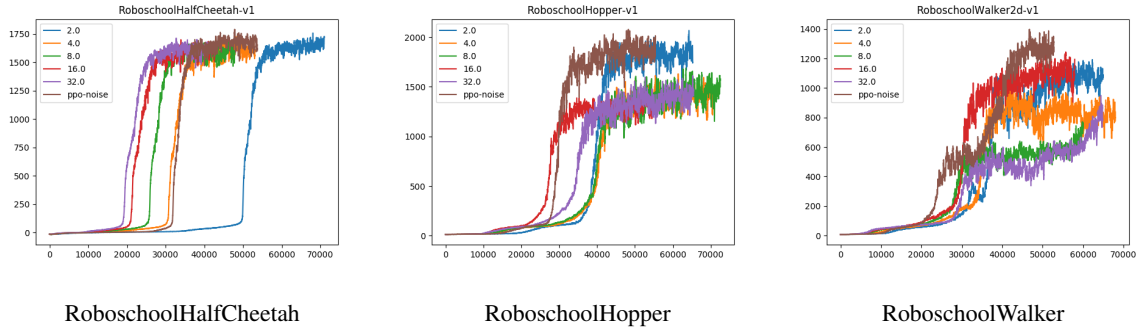
C.7.2. FOR  $\beta = 0.05$  AND  $stddev = 0.01$ 

Figure 17: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.05$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

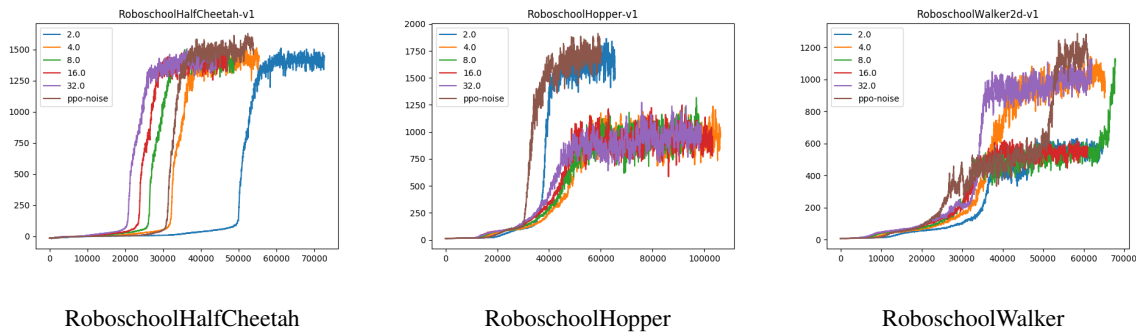
C.7.3. FOR  $\beta = 0.05$  AND  $stddev = 0.1$ 

Figure 18: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.05$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

C.7.4. FOR  $\beta = 0.05$  AND  $stddev = 1.0$

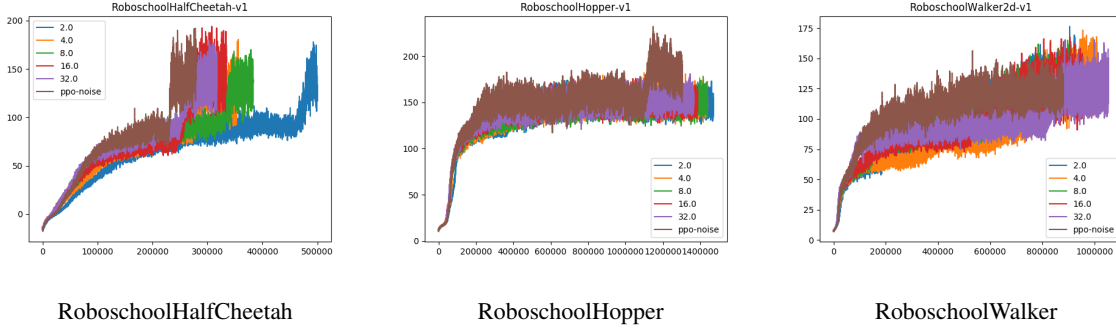


Figure 19: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.05$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

C.7.5. FOR  $\beta = 0.1$  AND  $stddev = 0.001$

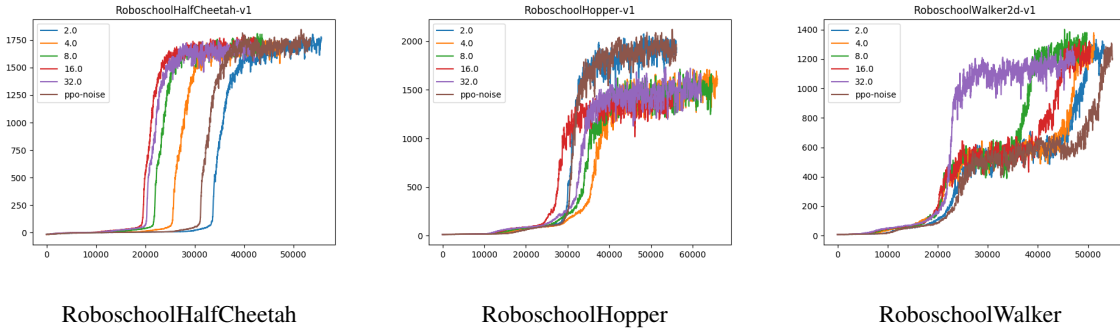


Figure 20: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

C.7.6. FOR  $\beta = 0.1$  AND  $stddev = 0.01$

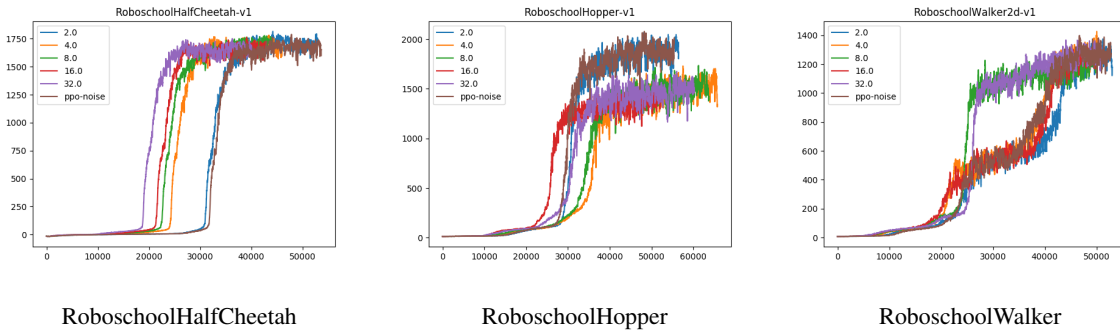


Figure 21: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with stadard deviation  $stddev$ .

C.7.7. FOR  $\beta = 0.05$  AND  $stddev = 0.1$

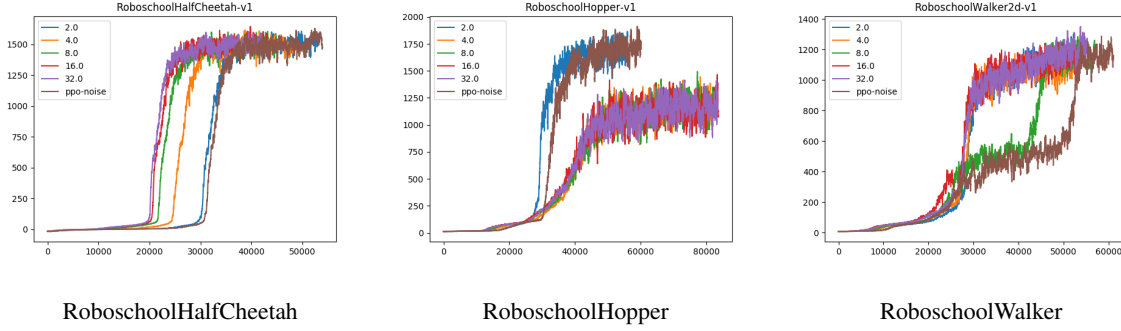


Figure 22: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with standard deviation  $stddev$ .

C.7.8. FOR  $\beta = 0.05$  AND  $stddev = 1.0$

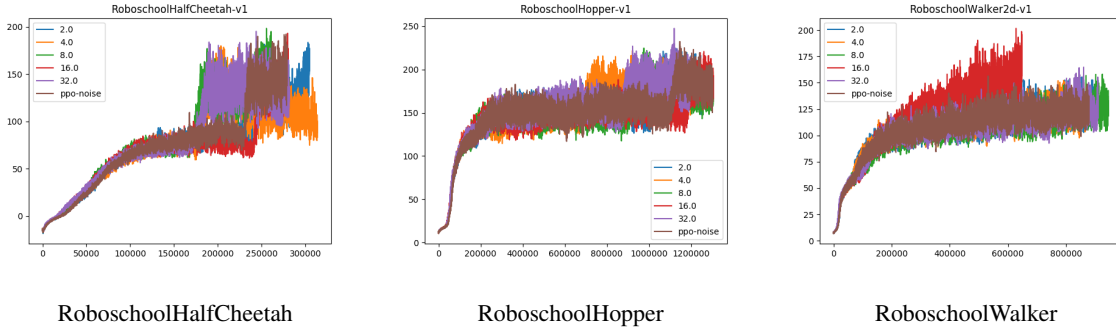


Figure 23: We run **ppo-original** and **ppo-len- $\gamma$ -dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original- $\gamma$ -dep-equalizer**. We keep  $\beta = 0.1$  and vary  $\alpha$ . But we also add gaussian noise to the observation, with standard deviation  $stddev$ .

## D. Study of RoboSchool environments underlying parameters

In the subsection C.3.2 we observe that while the models learn a policy, the episode length increases rapidly and surprisingly saturates, instead of increasing further. This slightly hampers the effect of analysis in length dependent trust region induced by  $\mathcal{D}_\gamma$ .

In the roboschool module, one can find [https://github.com/openai/roboschool/blob/master/roboschool/gym\\_forward\\_walker.py](https://github.com/openai/roboschool/blob/master/roboschool/gym_forward_walker.py) that the reward has 5 components:

1. alive
2. progress,
3. electricity-cost,
4. joints-at-limit-cost,
5. feet-collision-cost

The alive bonus has not been appropriately modulated to do not encourage the agent to die even though the agent can stay alive and collect more rewards. We tried to alter the *alive bonus* to make the staying alive a significant component for the

agnet. We multiply the alive bonus by a coefficient and study the agent behaviour. The alive bonus may be positive or negative, depending on specific parameters of the environment(for example, in the **humanoid** environment, if the center of mass of the robot falls below a certain limit, then it starts receiving negative alive bonuses, while alive bonuses are positive). The alive bonus for the environments is generally one unit. We multiplied the alive bonus with varying factors to observe the behavior of agent and see whether it learn to do not intentionally terminate the round (informally do not intentionally kill itself) as the model trains.

In the following plots, the x-axis represents the number of episodes seen by the model, and the y-axis represents the episode length. The numbers in the legend represent the factor by which the positive and negative alive bonus are scaled respectively. For example, the index 2.2\_2.2 means a factor of 2.2 (the first number) multiplies the positive alive bonus, and a factor of 2.2 (the second number) multiplies negative alive bonus. We empirically study the effect of this bonuses, but as one can see from the plots, the changes in alive bonus did not make significant change in the episode length. Fig. 24

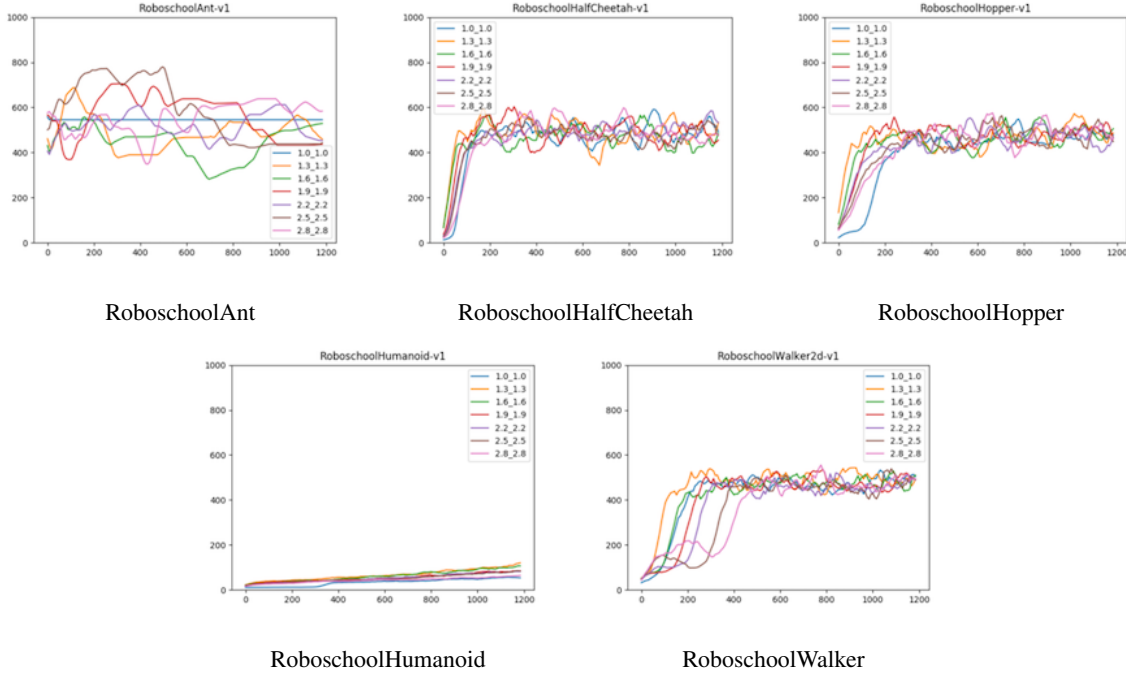


Figure 24: PPO behavior when we change the alive bonus. It seems that the agent does not learn to do not intentionally terminate the episodes. We believe the reward shaping deployed in PPO requires substantial study and critical modification to make them suitable for further studies.

## E. GTRPO

In this section we study the GTRPO behaviour on Robo-School environment . As it is mentioned in the Eq. 8 we need to train a network to estimate  $V$  and  $Q$  functions. We restate their definitions in the following;

$$\begin{aligned}
 V_{\pi}(y_h, h, y_{h-1}, a_{h-1}) &:= \\
 \mathbb{E}_{\pi} \left[ \sum_h^H \gamma^h r_h | y_h = y, y_{h-1} = y_{h-1}, a^{h-1} = a_{h-1} \right] \\
 Q_{\pi}(y_{h+1}, a, y_h, h) &:= \\
 \mathbb{E}_{\pi} \left[ \sum_h^H \gamma^h r_h | y_h = y, y_{h+1} = y_{h+1}, a^h = a \right]
 \end{aligned}$$

In practice we drop the  $h$  dependence. We train the  $V$  function using a simple neural network while we use samples of  $R(\tau)$  as the data. It is worth noting that we do not use Bellman residual methods to learn the  $V$  since there is not Bellman imposed structure when memoryless policies are acquired. For each tuple of  $y \rightarrow, a, \rightarrow y'$ , where arrows represent the ordering of the events, we train the on-policy  $V(y', y, a)$  to match the cumulative reward happens after observing  $y'$ . For the  $Q$  we deploy directly the sampled returns.

### E.1. Network design choice for $V$

We deploy a simple neural network with 2 fully connected layers to train for  $V$ . We try to tune for the number of nodes in each layers Fig. 25. In the legend of the plots, each tuple represents the number of nodes in first and second layer respectively.

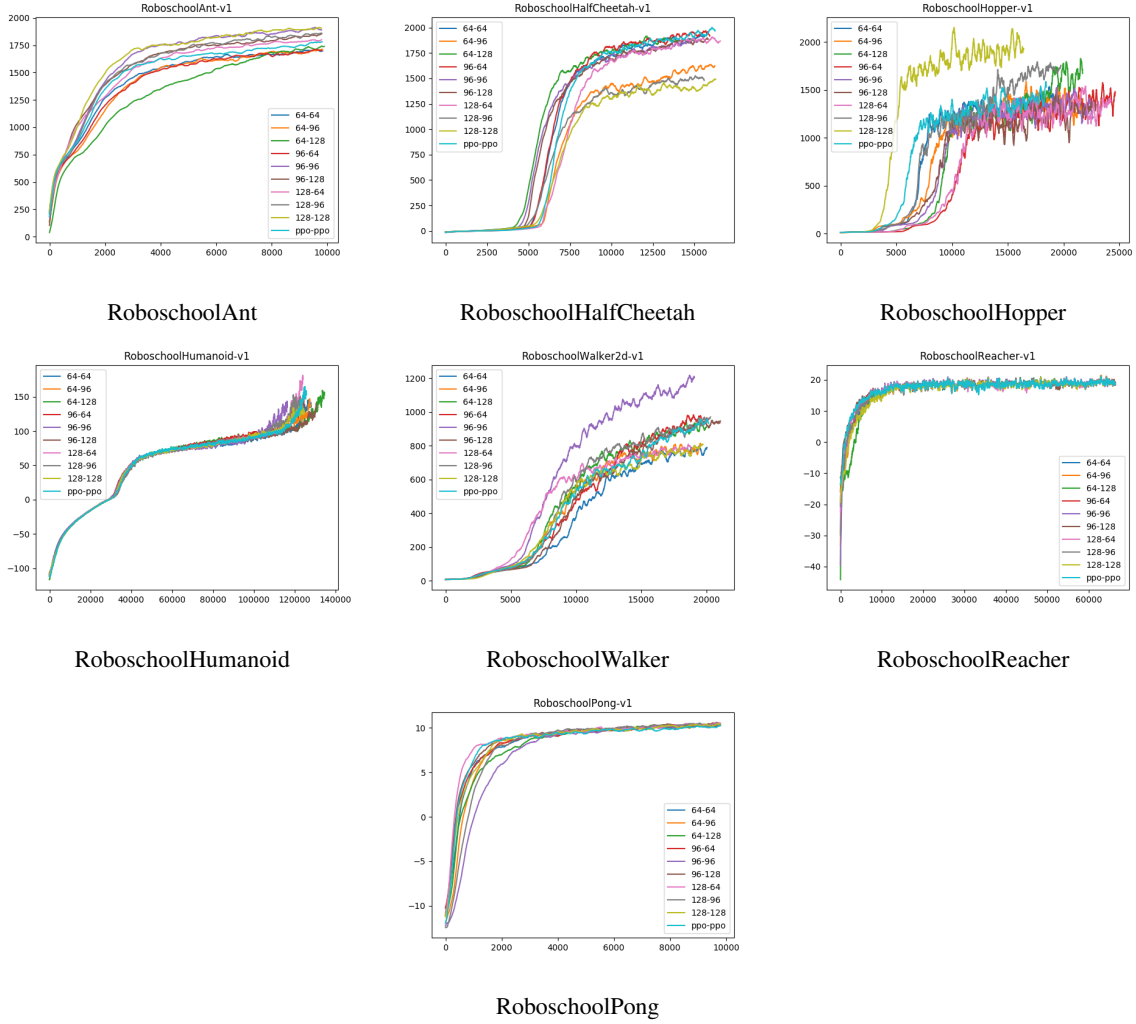


Figure 25: Performance of GTRPO when we use different design choice to train the  $V$ .

## E.2. Plot with variance

We observed that the neural net with nodes 96-96 performed consistently in comparison to its contemporaries Fig. 26. Following are the comparison plots along with variance.

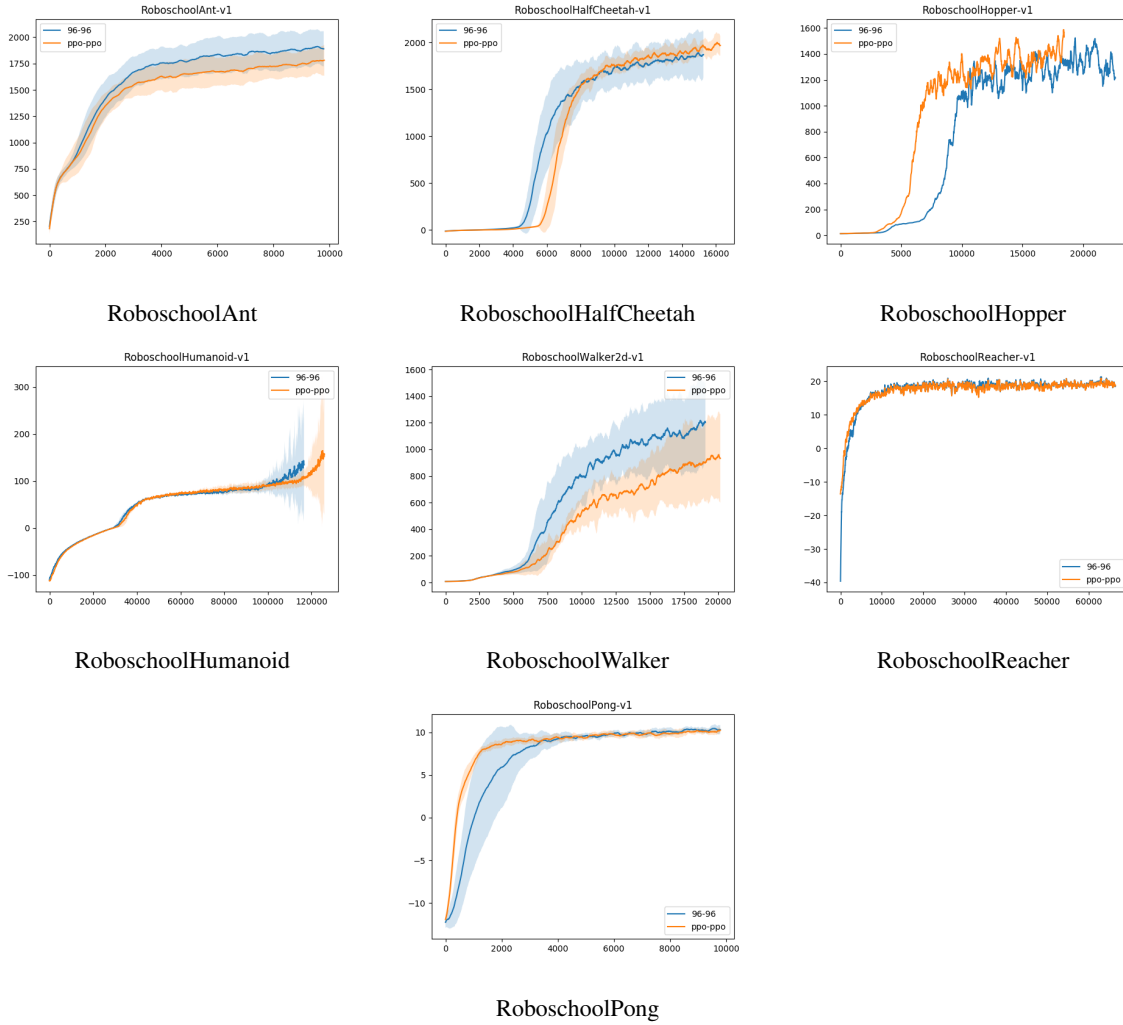


Figure 26: Performance of GTRPO when  $V$  is approximated with two layer neural network of size 96-96.

### E.3. Noise

For further study, we also introduced Gaussian noise into the observation of RoboSchool environments and reduce the 'observability' of the states. We report the performance for variety of noise levels.

#### E.3.1. STDDEV = 0.001

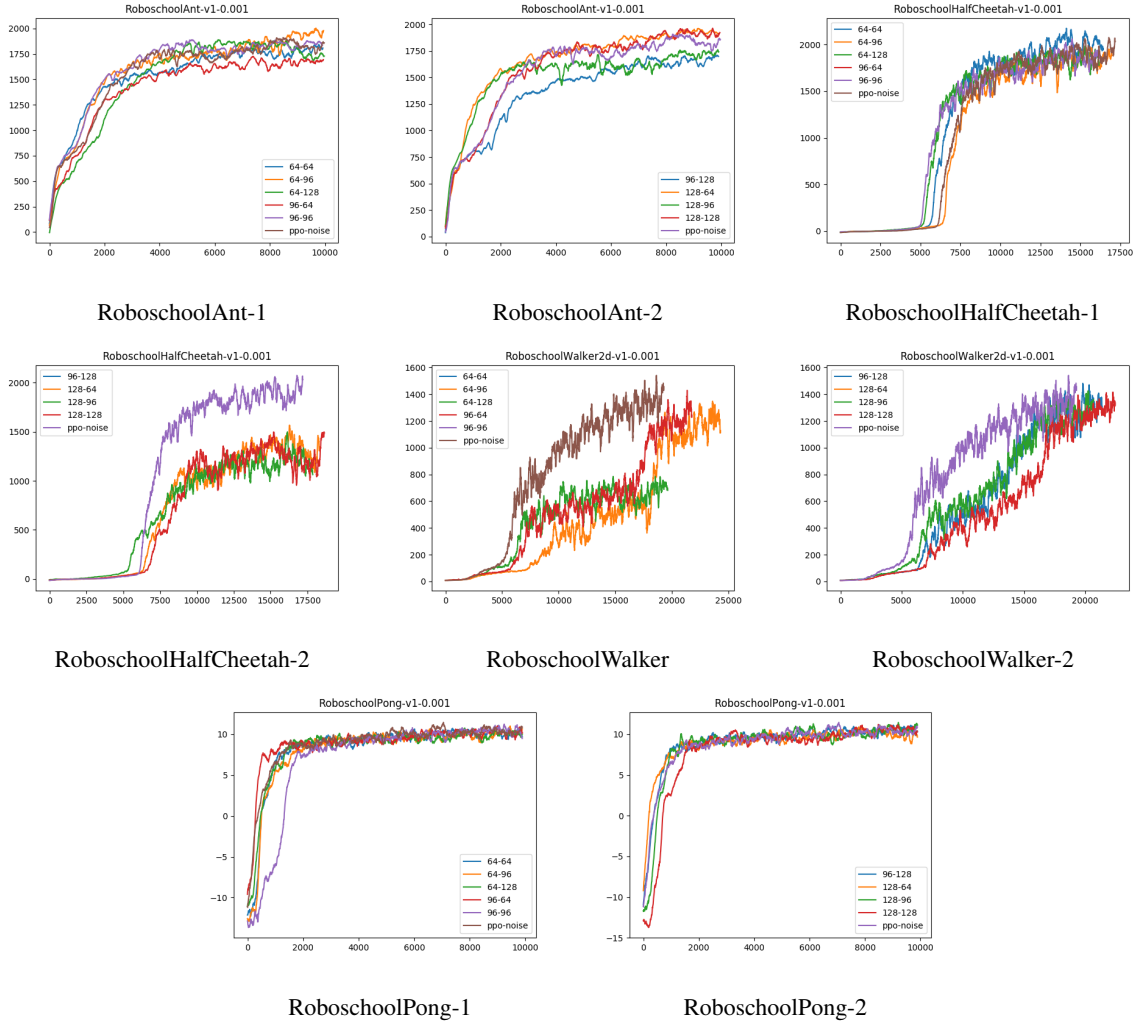


Figure 27: Behavior of GTRPO under noised observation with standard deviation of 0.001



E.3.2. STDDEV = 0.01

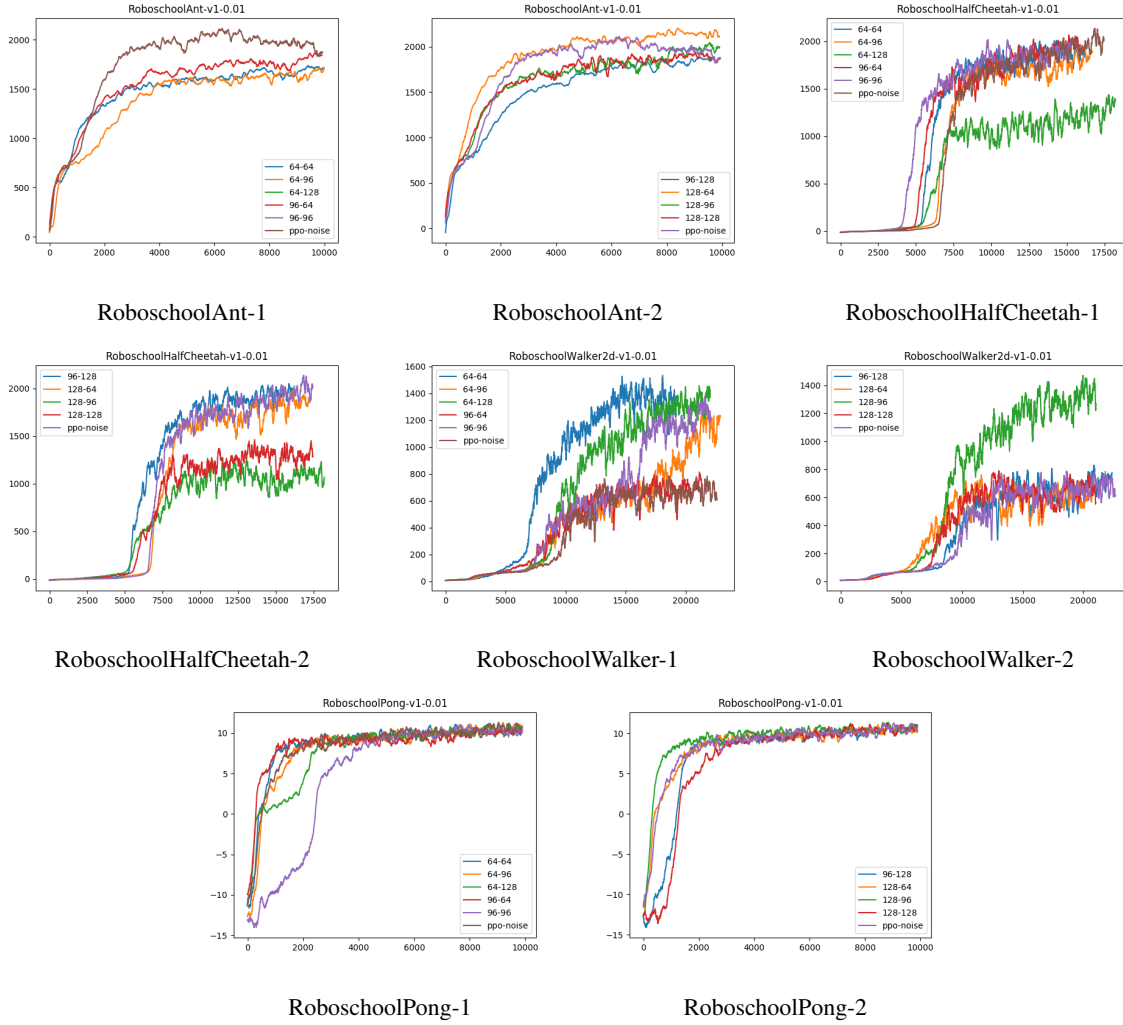


Figure 28: Behavior of GTRPO under noised observation with standard deviation of 0.01

## E.3.3. STDDEV = 0.1

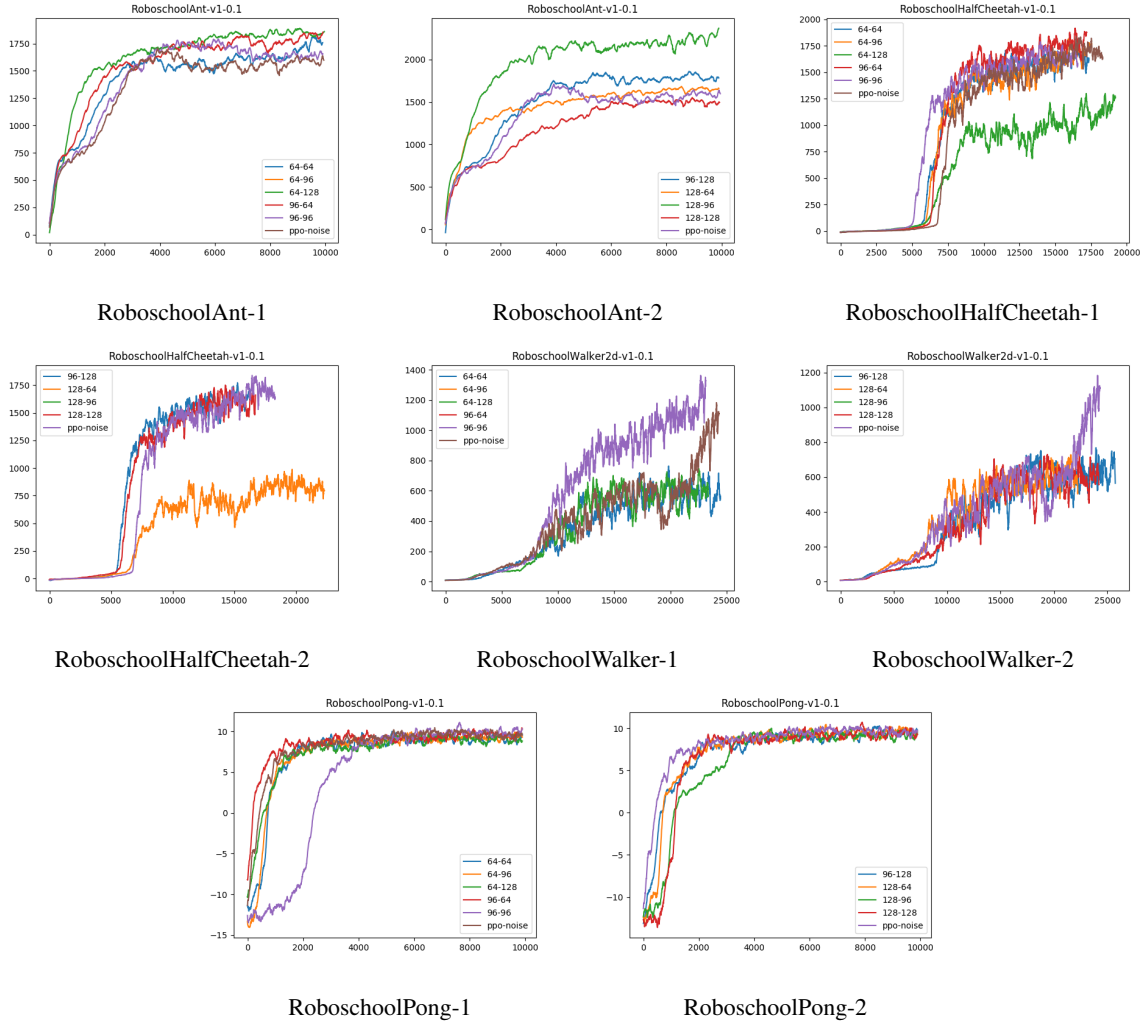


Figure 29: Behavior of GTRPO under noised observation with standard deviation of 0.1

## E.4. stddev = 1.0

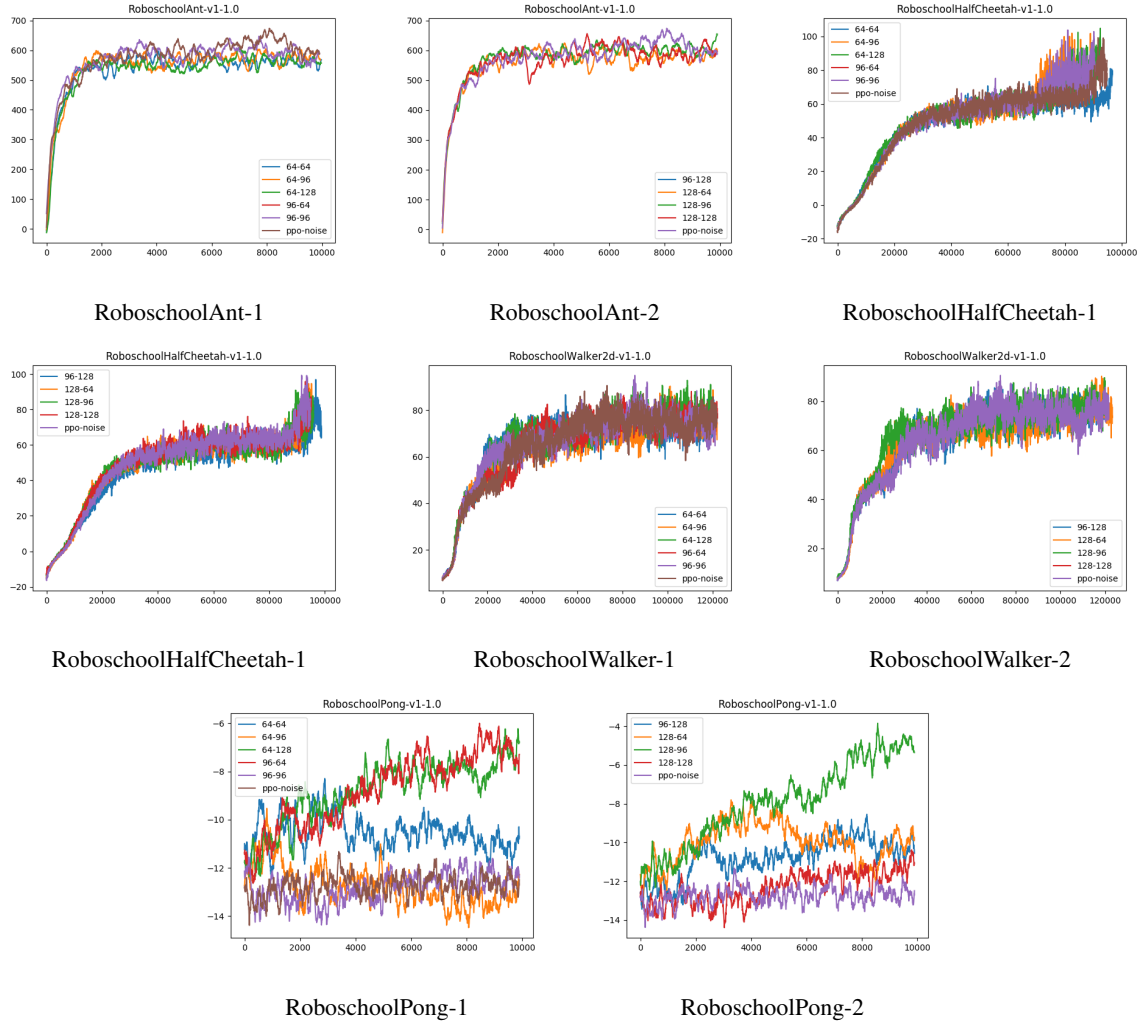


Figure 30: Behavior of GTRPO under noised observation with standard deviation of 1.0

E.4.1. STDDEV = 10.0

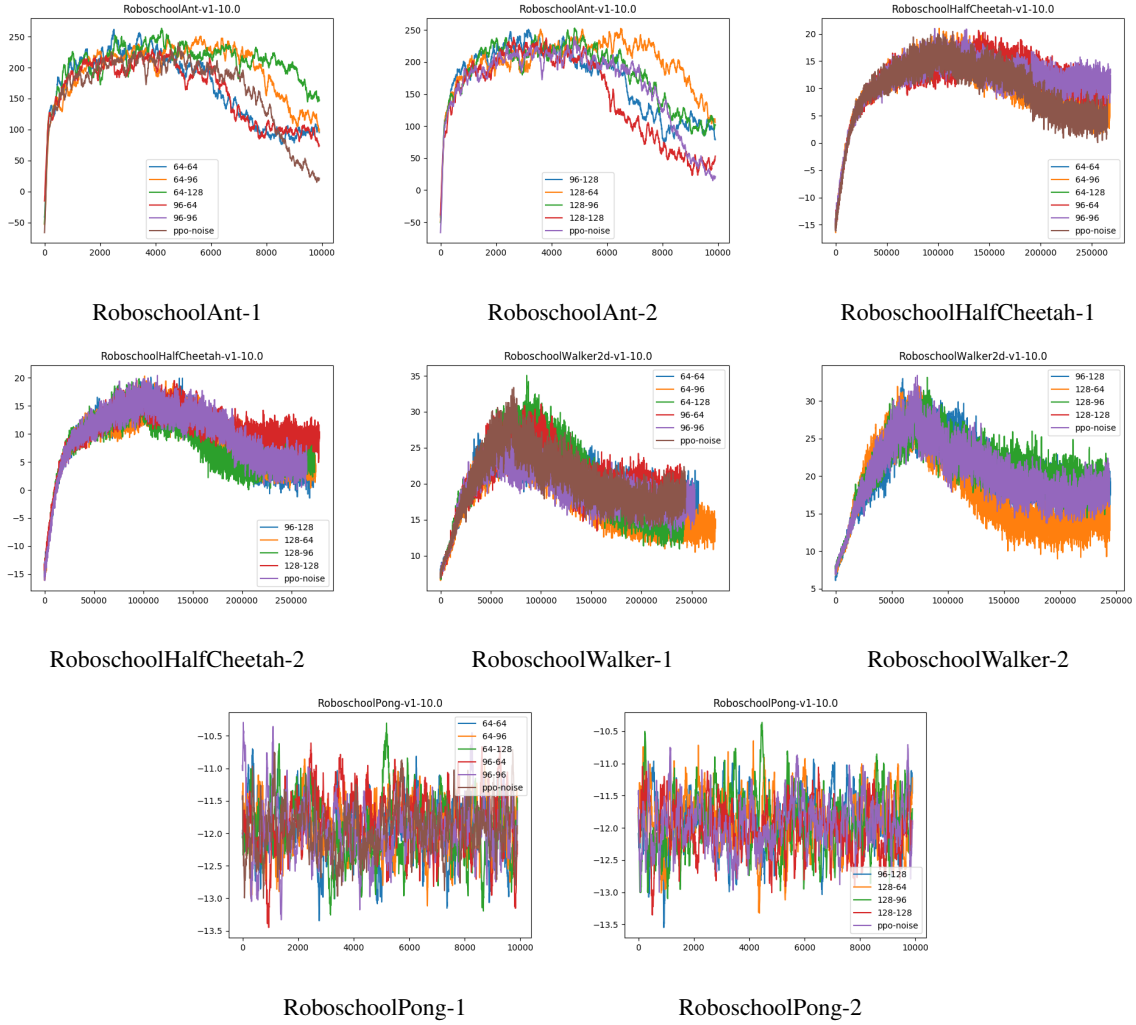


Figure 31: Behavior of GTRPO under noised observation with standard deviation of 10.0

## F. PPO through signSGD

We made a further empirically study of a different way of imposing the trust region. signSGD, as a optimization method computes the gradient vector but move in the direction of sign of gradient. This approach implicitly prevents big changes in the parameter space and moves the parameters in all the directions with the same magnitude. It also forces to move with the same magnitude toward the directions that the trust region suggests low changes in them.

signSGD:

- $g_k \leftarrow$  stochastic gradient
- $x_{k+1} \leftarrow x_k - \delta \text{sign}(g_k)$

We apply PPO on the Robo-school environment and deploy the signSGD optimizer for the policy gradient.

The legend in the following plots denote the *learning rate* that we try for signSGD.

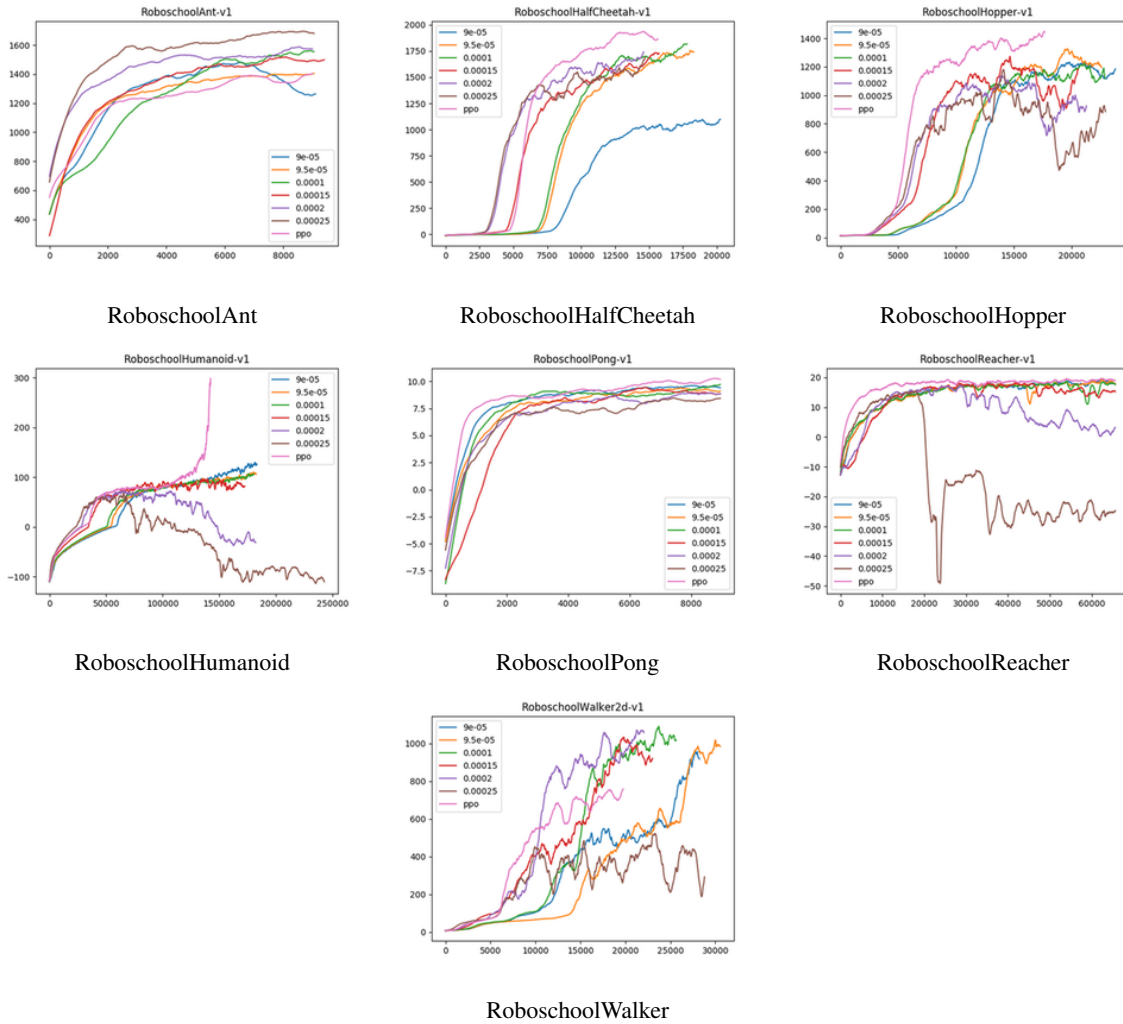


Figure 32: Behavior of PPO when signSGD is deployed as the optimizer